THE GRAPHICAL KERNEL SYSTEM (GKS)

VIEWGRAPHS FOR A TUTORIAL

D A DUCE and D C SUTCLIFFE

on

11 February 1987

Science and Engineering Research Council

Informatics Division
Rutherford Appleton Laboratory
Chilton, Didcot, Oxon, OX11 0QX

GRAPHICS STANDARDS

VIEWGRAPHS

F R A HOPGOOD and D A DUCE

Science and Engineering Research Council

Informatics Division
Rutherford Appleton Laboratory
Chilton, Didcot, Oxon, OX11 0QX, UK

May 1986

OUTLINE

PART A

Introduction, History, Concepts
Output, Transformations } 3 hrs

Attributes
Logical Input } 2 hrs

PART B

Segments
Workstations
Environment } 3 hrs

Input, Metafiles } 2 hrs

Standardisation Process
Future Standards
- GKS-3D
- PHIGS
- CGI
- CGM
- WM

} 1 hr +

COURSE BOOK

Hopgood Duce
Gallop Sutcliffe

'Introduction to the Graphical Kernel System- GKS'

Academic

Second Edition
Paperback

Currently only book that agrees with International Standard

PART A Chapters 1, 2, 3, 5, 6

PART B Chapters 4, 7, 8, 9, 10, 11

Recommended

Bono 'A survey of Graphics Standards and their role in Information Interchange' IEEE COMPUTER OCTOBER 1985

OTHER BOOKS

(nearly correct)

Enderly, Kansy, Pfaff

Computer Graphics Programming
GKS - The Graphics Standard
SPRINGER (being updated 1986)

Sproull, Sutherland, Ullner

Device Independent Graphics
Mc GRAW HILL

Hearn, Baker

Computer Graphics
PRENTICE HALL

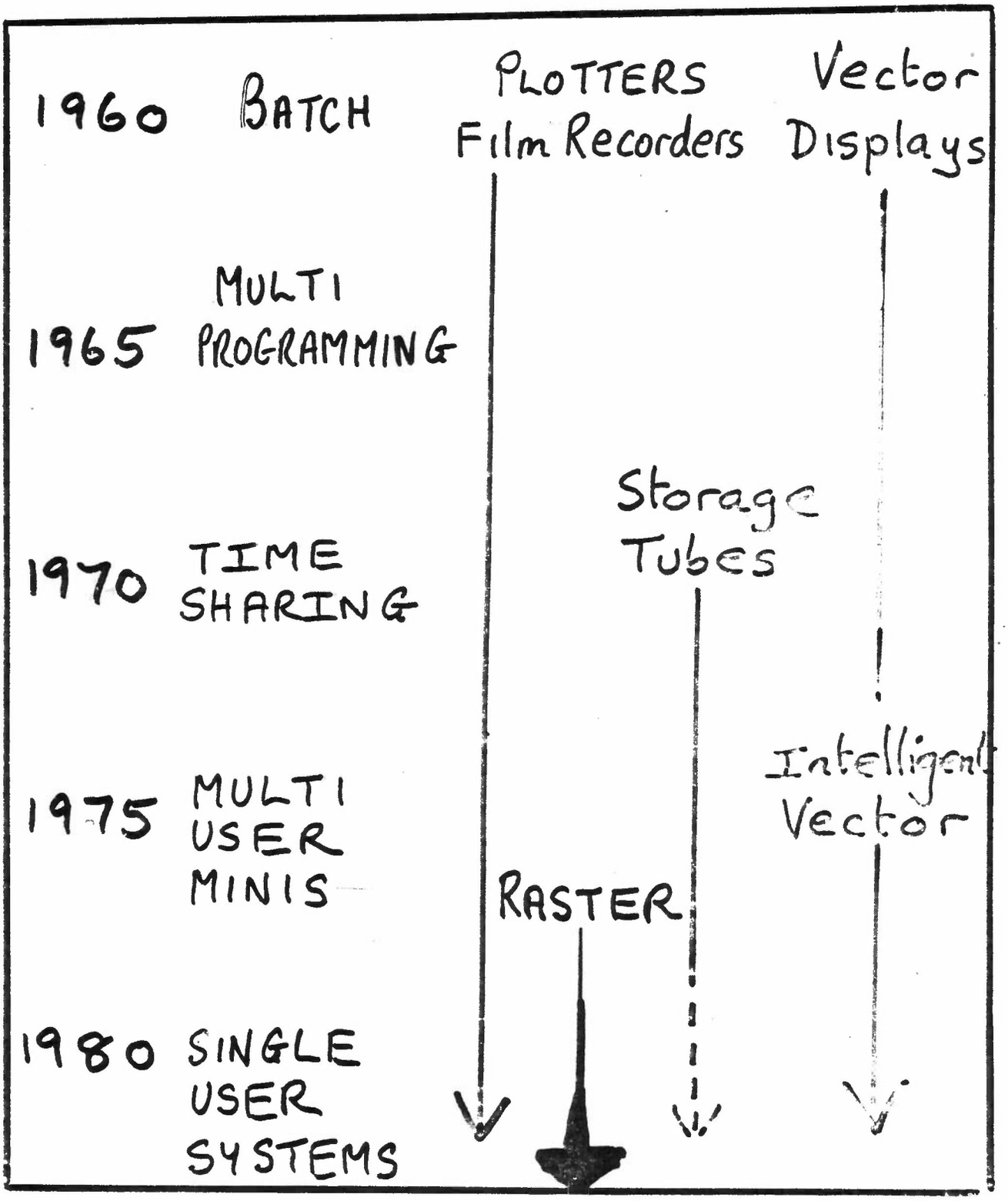
Brown

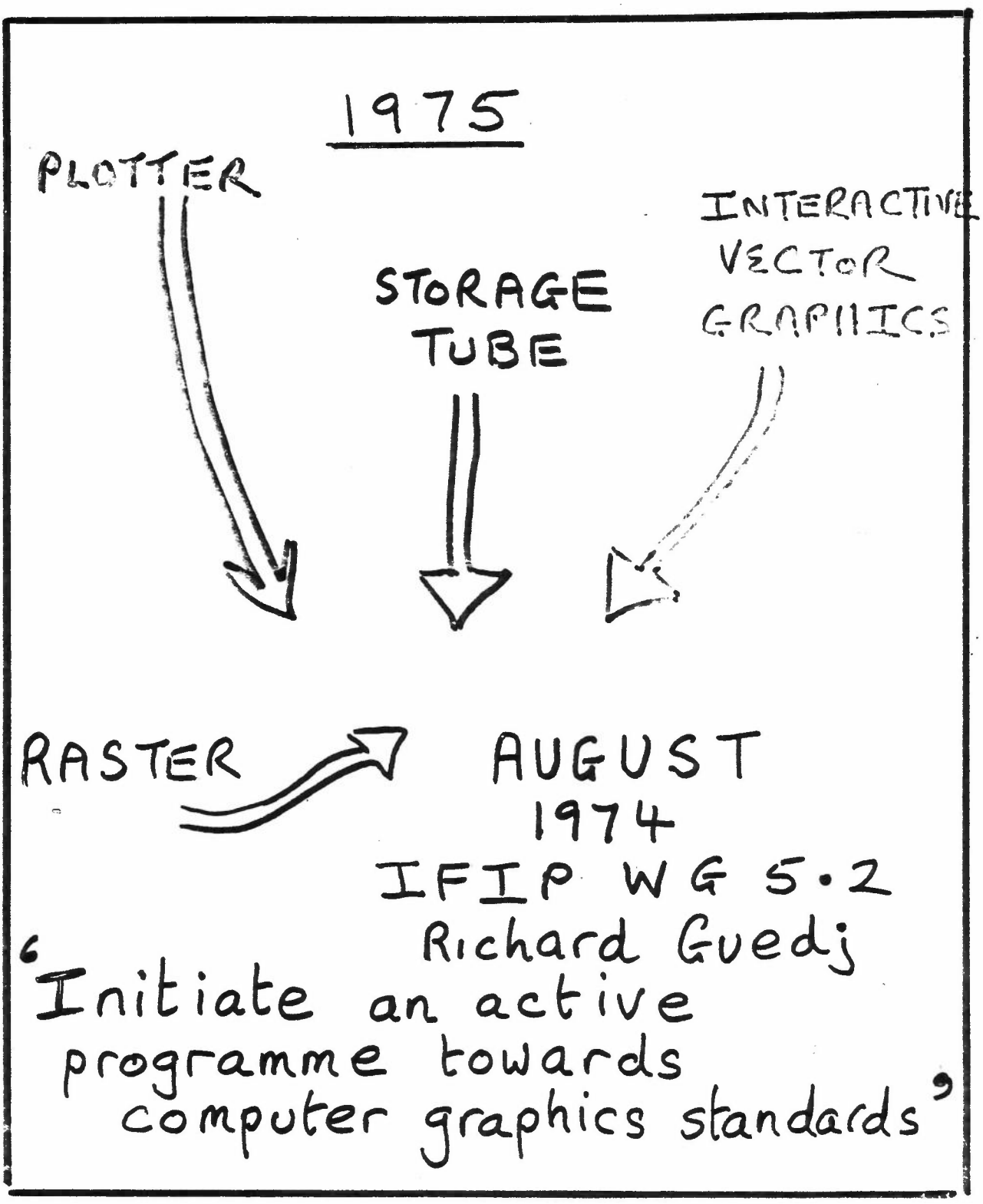
Understanding PHIGS
TEMPLATE

GKS TUTORIAL

INTRODUCTION

- Early History
- Goals, Principles, Concepts
- Standards History
- GKS Concepts


1960 BATCH
PLOTTERS
Film Recorders
Vector
Displays
1965 MULTI PROGRAMMING
Storage
Tubes
1970 TIME SHARING
Intelligent
Vector
1975 MULTI USER MINIS
RASTER
1980 SINGLE USER SYSTEMS


1975
PLOTTER
STORAGE TUBE
INTERACTIVE VECTOR GRAPHICS
RASTER
AUGUST 1974
IFIP WG 5.2
Richard Guedj
'Initiate an active programme towards computer graphics standards'

EARLY HISTORY

1964-1974	Local de facto standards Device specific software
1974	AGM SIGGRAPH GSPC formed Graphics Standards Planning Committee
1974	IFIP WG 5.2 – Richard Guedj

IFIP WG 5.2

August 1974 Malmö Sweden

IFIP WG 5.2 invited Richard Guedj to initiate an active programme directed towards establishing standards for computer graphics

May 1976 - SEILLAC I

Workshop of Experts to try and define fundamental Methodology of Graphics

[Proceedings : Methodology in Computer Graphics - North Holland]

REQUIREMENTS OF A STANDARD

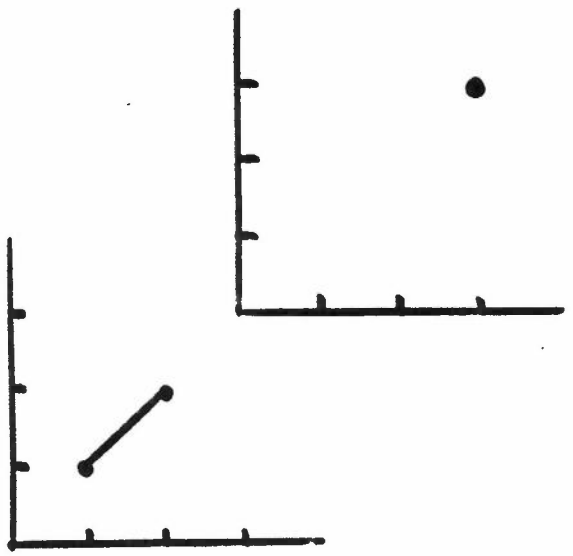
- Meets needs of most users
- Independent national/commercial interests
- Defined by experts
- Rationalise current practice - but provide significant advance
- Improve portability of software
- Passive and Interactive
- Not conflict with other standards
- Suitable for range of devices
- First of a set of compatible standards

ISSUES

- What functions - raster ?
- Which application areas ?
- What level for first standard ?
- How to achieve device independence ?
- Interaction of Current Position and coordinate transformation

MEANING OF CURRENT POSITION

```
TOXY (1, 1)
DRAW (2,2)
TRANS (3,3)
DRAW (3,3)
```



From where is second line drawn

Possibilities : Undefined

(0, 0)

(-1,-1)

(2, 2)

MODELLING and VIEWING

- Coordinate transformations used to :

  (1) VIEW a picture

  (2) COMPOSE or MODEL picture

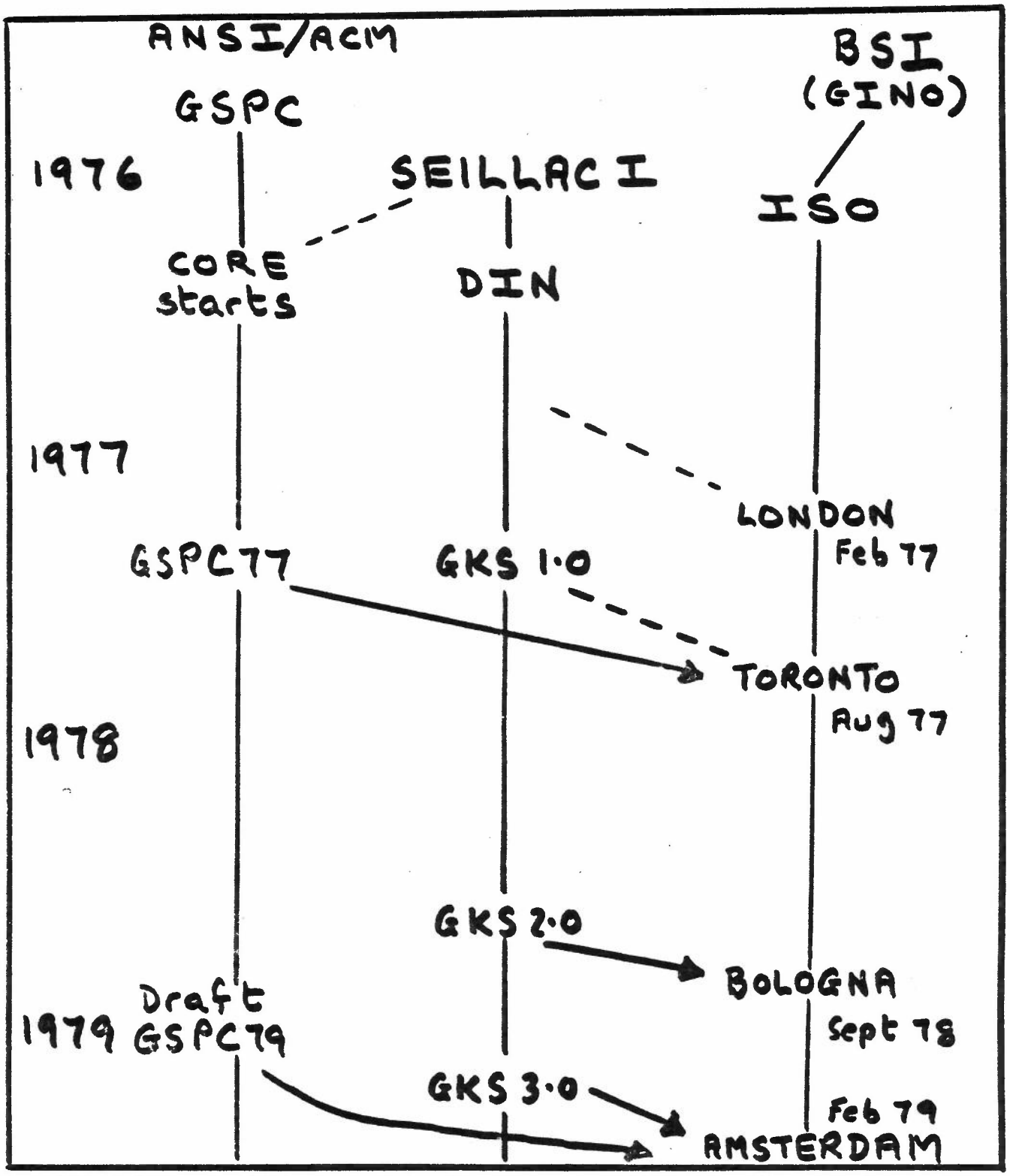
- Many packages in 1974 used same function, both tasks

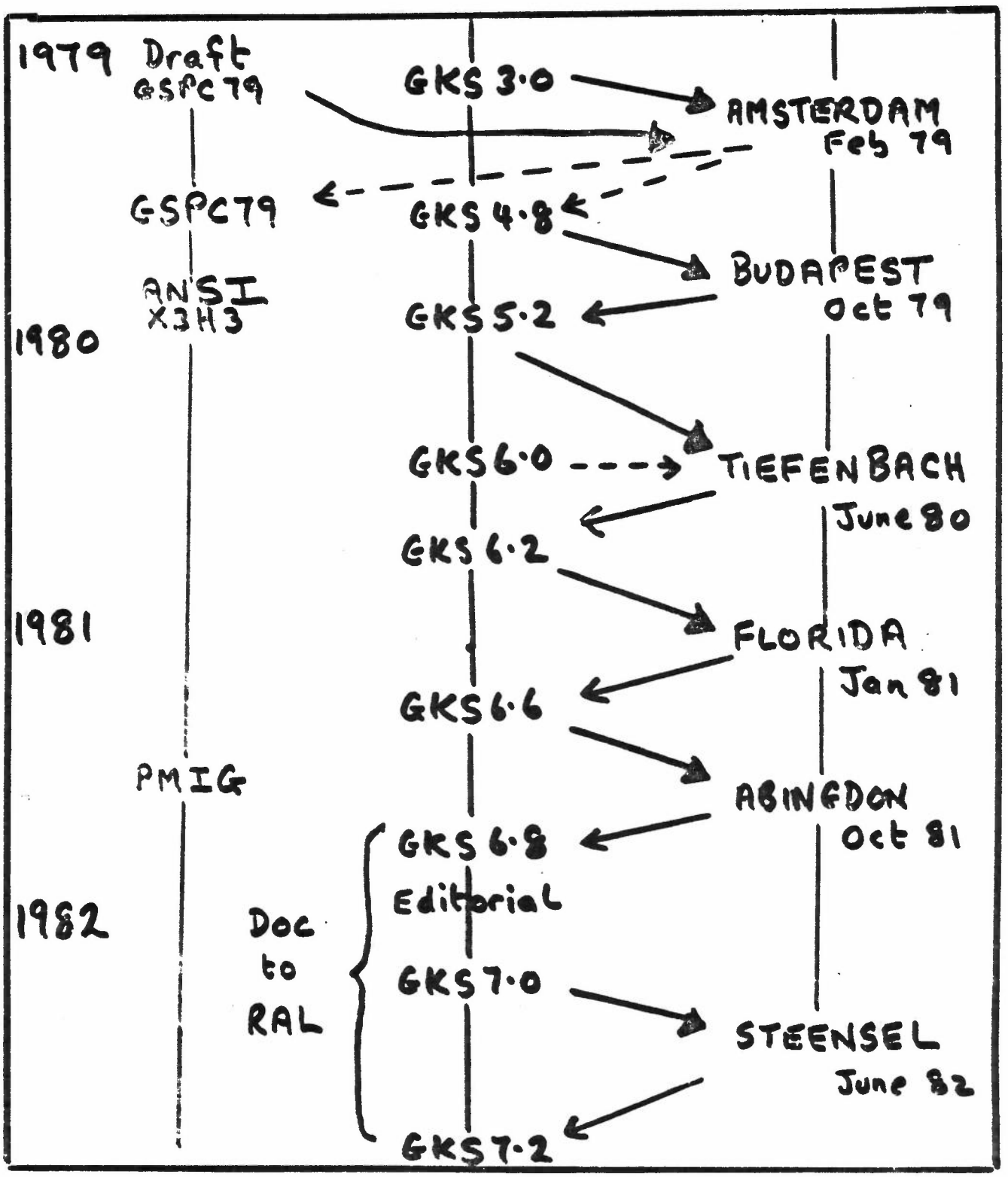
- Current Position concept emphasised the problem

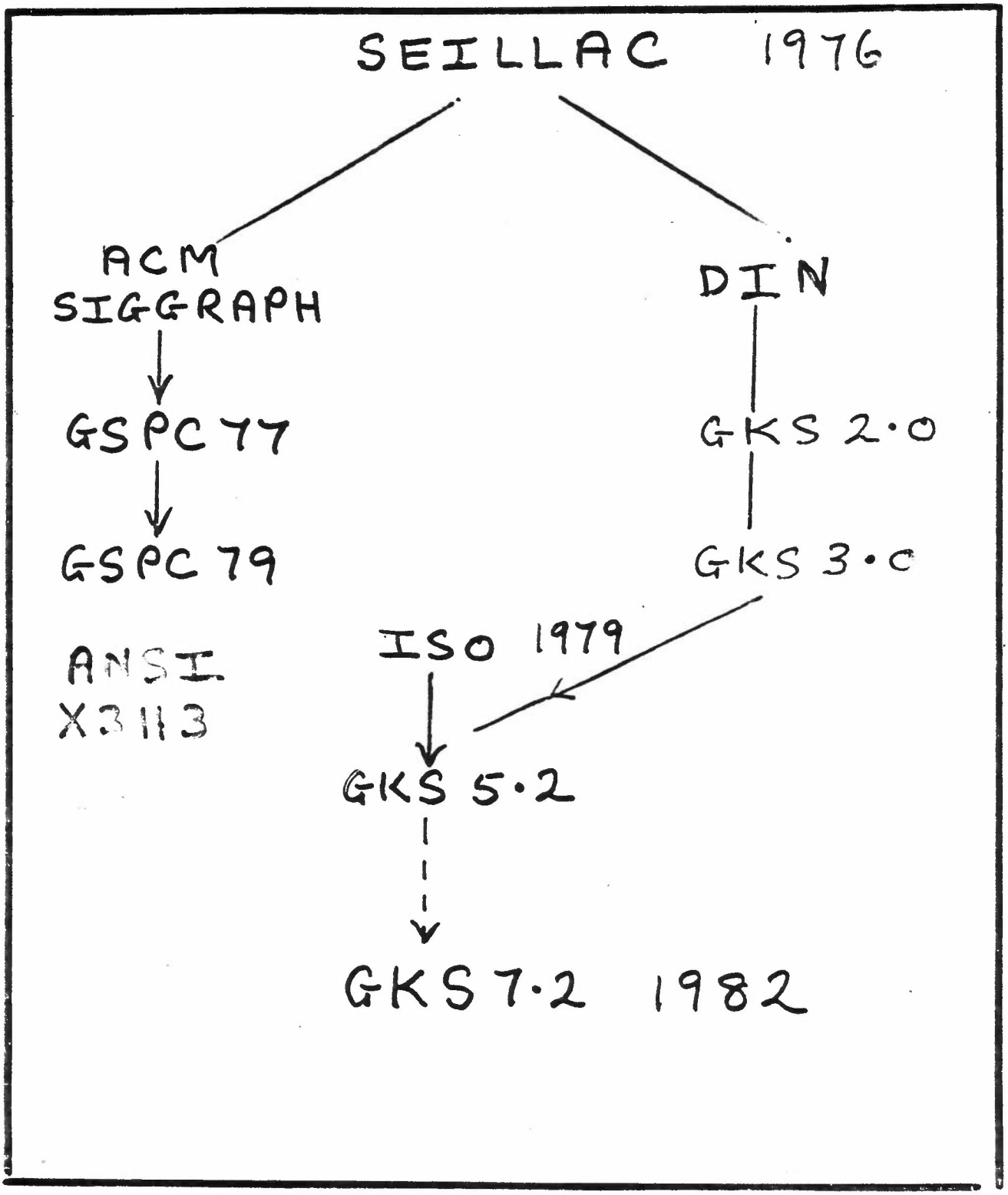
SEILLAC RECOMMENDATIONS

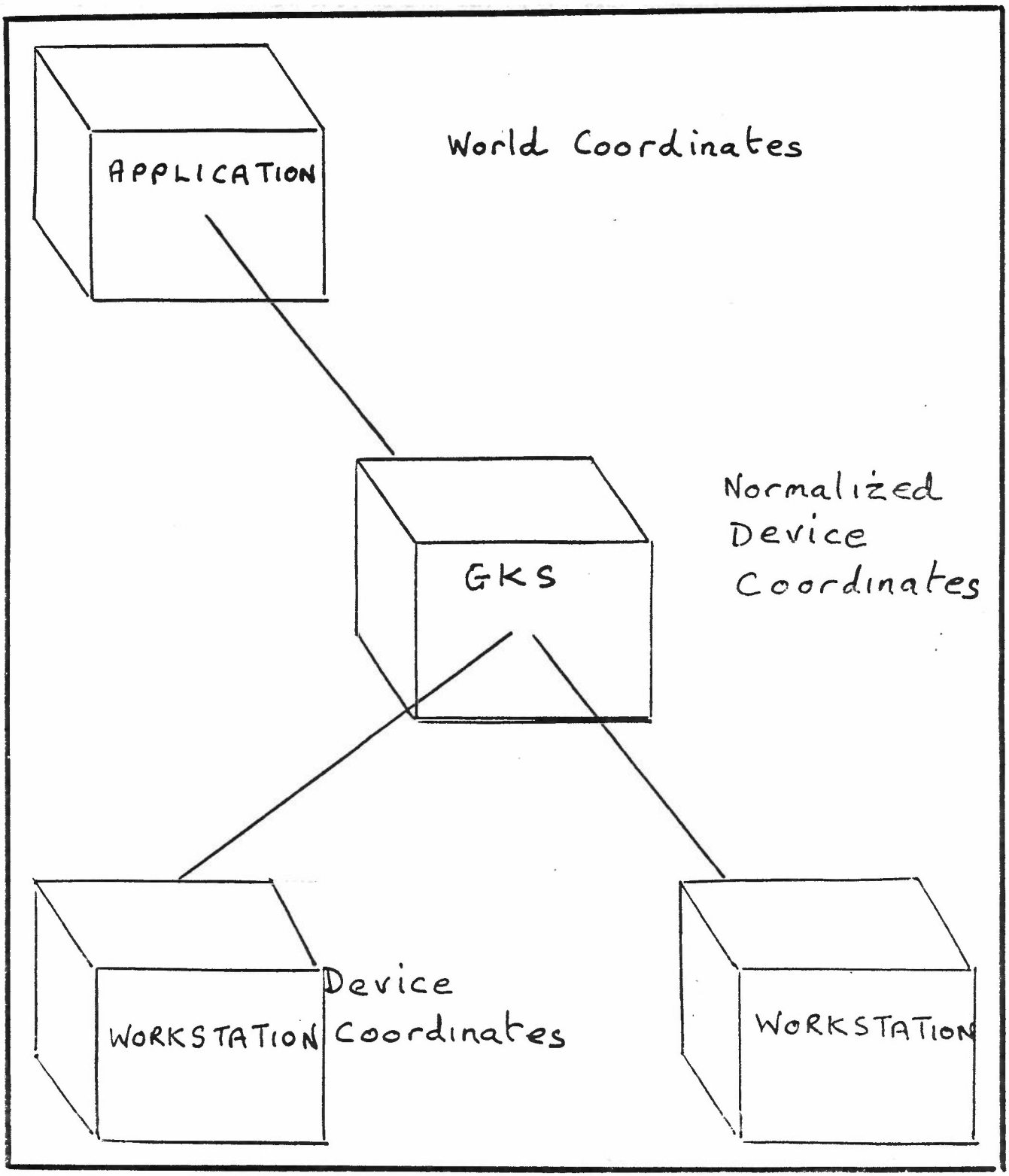
- Clear distinction needed between graphics subsystem (CORE or KERNEL) and modelling systems built on top
- Current systems confuse use of coordinate transformations for viewing and modelling
- GRAPHICS SUBSYSTEM should concentrate on viewing aspects of picture already constructed in world coordinates
- GRAPHICS SUBSYSTEM should be first standard activity

  IT SHOULD DO NO MODELLING

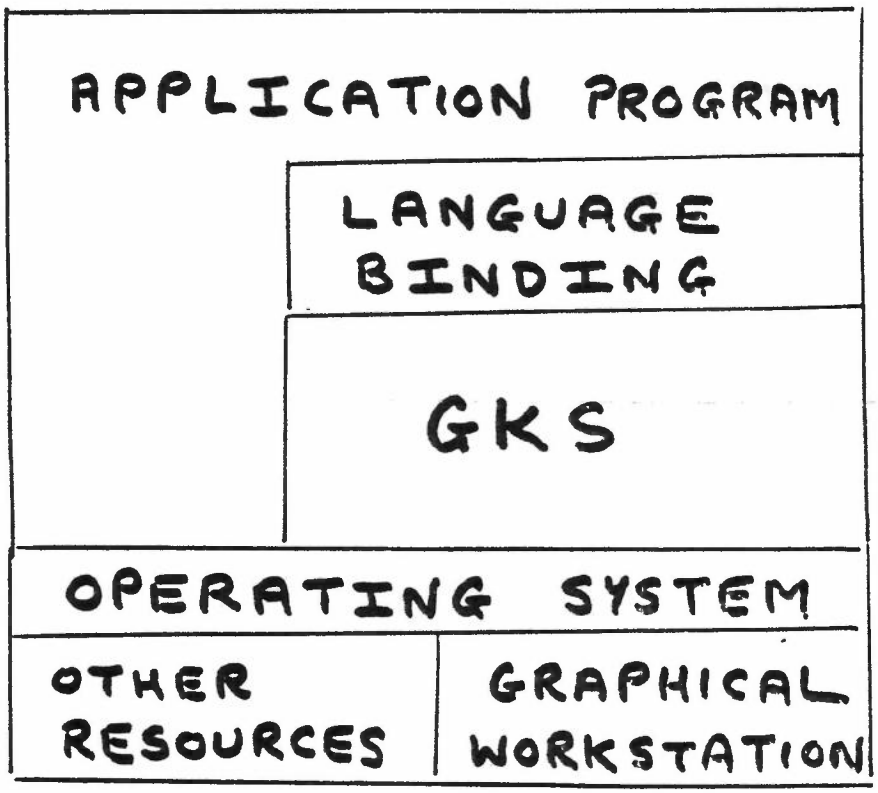

ANSI/ACM
BSI
(GINO)
GSPC
1976
SEILLAC I
ISO
CORE
starts
DIN
1977
LONDON
Feb 77
GSPC77
GKS 1.0
TORONTO
Aug 77
1978
GKS 2.0
BOLOGNA
Sept 78
Draft
1979 GSPC79
GKS 3.0
Feb 79
AMSTERDAM

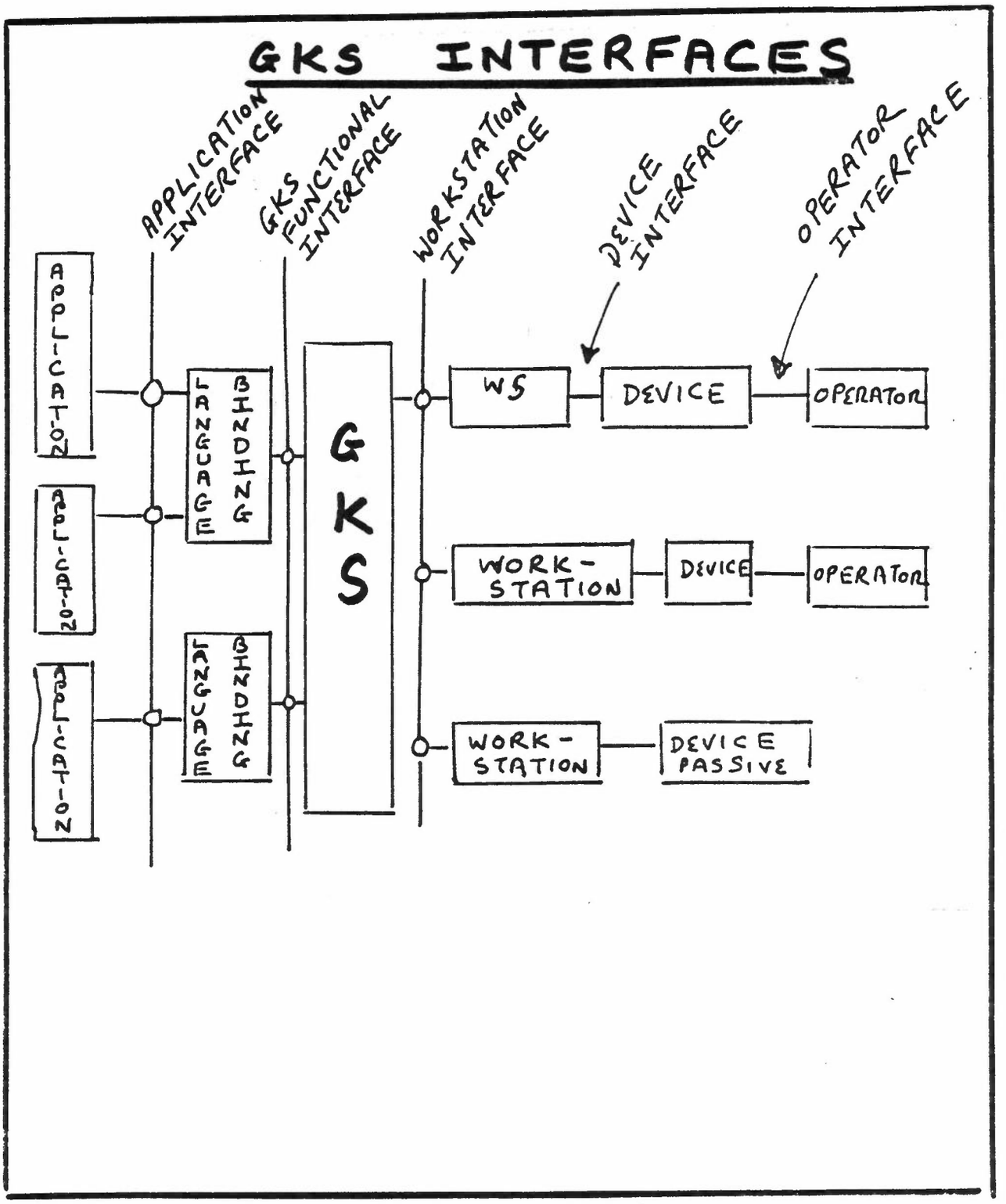

1979
Draft GSPC 79
GKS 3.0
AMSTERDAM Feb 79
GSPC 79
GKS 4.8
BUDAPEST Oct 79
ANSI X3H3
1980
GKS 5.2
GKS 6.0
TIEFENBACH June 80
GKS 6.2
1981
FLORIDA Jan 81
GKS 6.6
PMIG
ABINGDON Oct 81
GKS 6.8
Editorial
1982
Doc to RAL
GKS 7.0
STEENSEL June 82
GKS 7.2

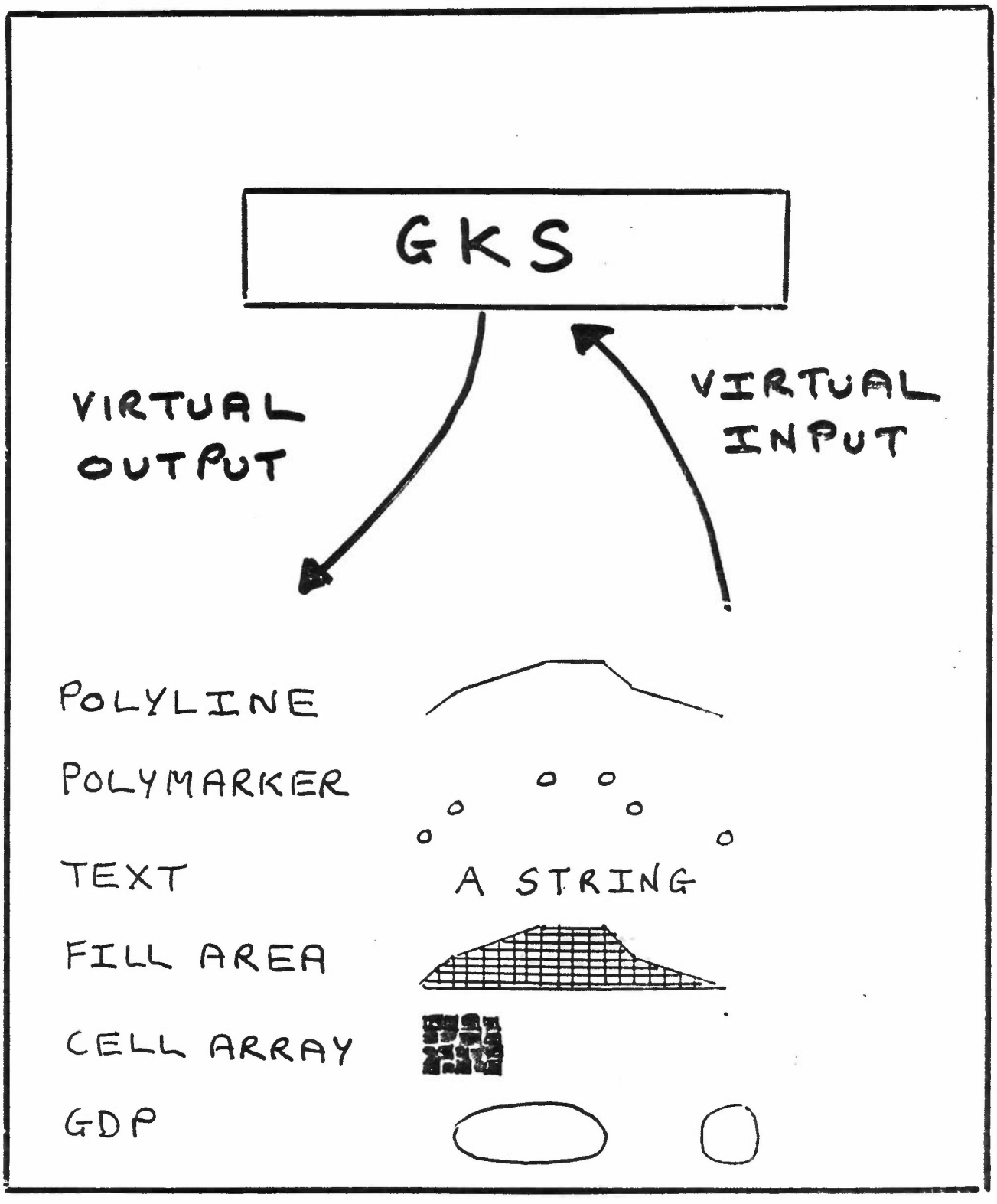

SEILLAC 1976
ACM SIGGRAPH
GSPC 77
GSPC 79
DIN
GKS 2.0
GKS 3.0
ANSI X3113
ISO 1979
GKS 5.2
GKS 7.2 1982

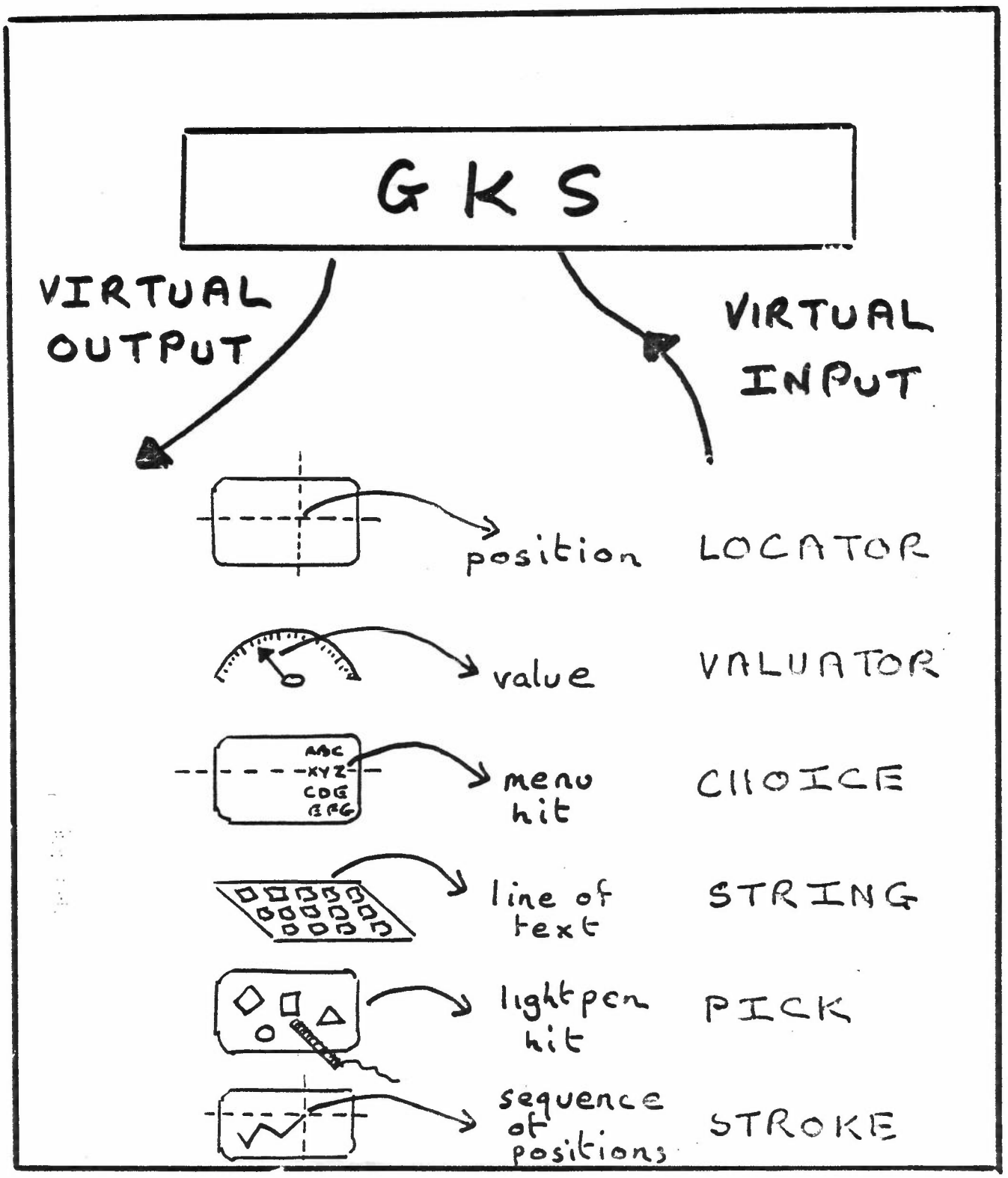

APPLICATION
World Coordinates
GKS
Normalized
Device
Coordinates
WORKSTATION
Device
Coordinates
WORKSTATION

LAYER MODEL OF GKS




GKS INTERFACES
APPLICATION INTERFACE
GKS FUNCTIONAL INTERFACE
WORKSTATION INTERFACE
DEVICE INTERFACE
OPERATOR INTERFACE
APPLICATION
APPLICATION
APPLICATION
LANGUAGE BINDING
LANGUAGE BINDING
GKS
WS
DEVICE
OPERATOR
WORK-STATION
DEVICE
OPERATOR
WORK-STATION
DEVICE PASSIVE


GKS
VIRTUAL OUTPUT
VIRTUAL INPUT
POLYLINE
POLYMARKER
TEXT
A STRING
FILL AREA
CELL ARRAY
GDP


GKS
VIRTUAL OUTPUT
VIRTUAL INPUT
position
LOCATOR
value
VALUATOR
ABC
XYZ
CDG
BFG
menu hit
CHOICE
line of text
STRING
lightpen hit
PICK
sequence of positions
STROKE

GKS AIMS

- Completeness
- Minimality
- Compactness
- Richness
- Robustness/ Error Handling
- Language Independent
- Efficiency
- Device Independence
- Device Richness

GKS BASIC CONCEPTS

- Output Primitives
- Attributes
- Workstations
- Coordinate Transformations
- Input
- Segments
- Metafiles
- 2D Only
- Levels

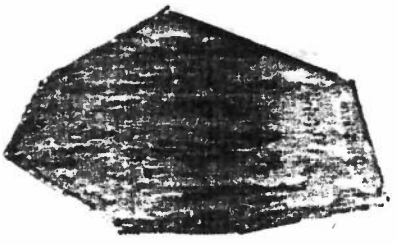
GKS OUTPUT PRIMITIVES

Polyline

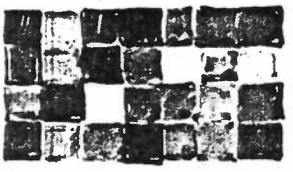
Polymarker

Text characters

Fill Area



Cell Array



GDP

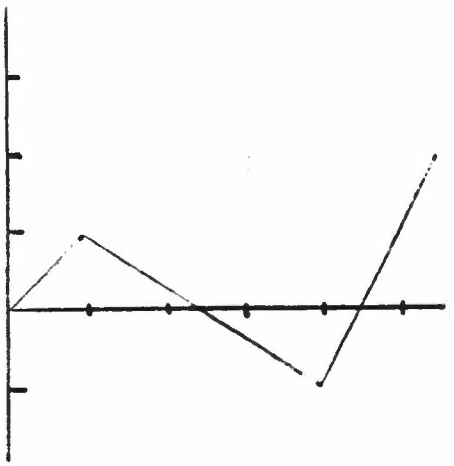
POLYLINE (n, list of points)

CALL GPL (N, XA, YA)

Polyline consists of N-1 line segments joining :

first point to second
second point to third
- - - - - - - - - -
last but one to last

POLYLINE EXAMPLE



N	XA	YA
1	0	0
2	1	1
3	4	-1
4	5	2

MAJOR POINTS

- No Current Position
- Absolute World Coordinates
- No single line primitive

SINE CURVE

GKS

```
      REAL XA(21), YA(21)

      DO 100 J=1,21
      XA(J) = (J-1) * 0.1 * PI
      YA(J) = SIN ( XA(J) )
100   CONTINUE

      POLYLINE(21, XA, YA)
```

EXISTING PACKAGE

```
      MOVE (0, 0)

      DO 100 J=2,21
      X = (J-1) * 0.1 * PI
      TO (X, SIN(X))
100   CONTINUE
```

POLYLINE ASPECTS

The set of points describe the path of the polyline

HOW the polyline is drawn is controlled by the

aspects

of the polyline

- Linetype
- Linewidth Scale Factor
- Colour Index

[the method of setting these aspects will be described later]

LINETYPE

< 0	implementation dependent	
0	invalid	
1	solid	———————
2	dashed	— — — —
3	dotted	. . . . .
4	dashed-dot	— . — . — . — . — .
⩾ 5	subject to registration	

LINEWIDTH SCALE FACTOR

5.0		▬▬▬▬▬▬▬▬
1.0	standard	———————
0.5		———————

Parameter defines linewidth as a function of standard linewidth

COLOUR INDEX

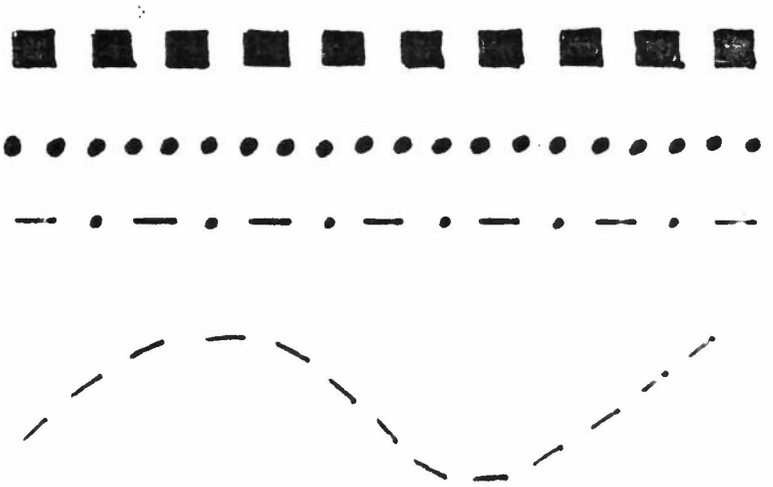
1 ———————————

2 ———————————

3 ———————————

Colour Index points to a colour description which can be varied

Polylines can have all aspects specified :



BACKGROUND COLOUR

- Entry 0 in the Colour Table defines the background colour
- This can be altered :

  raster : change background colour

  plotter : change colour of paper
- If background colour cannot be changed, attempts to change it will be ignored.

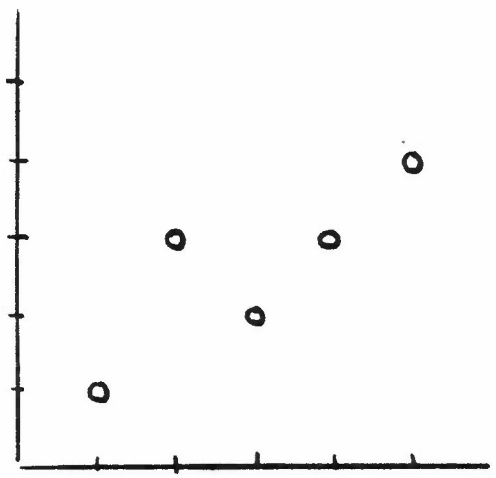
POLYMARKER (n, list of points)

CALL GPM (N, XA, YA)

The set of points specified are marked by a marker

The marker has a well-defined centre and is the same shape for each point marked by the polymarker

POLYMARKER EXAMPLE



N	XA	YA
1	1	1
2	2	3
3	3	2
4	4	3
5	5	4

POLYMARKER ASPECTS

The set of points define the positions of the markers

HOW the markers are drawn is controlled by the aspects of the polymarker

- Markertype
- Markersize Scale Factor
- Colour Index

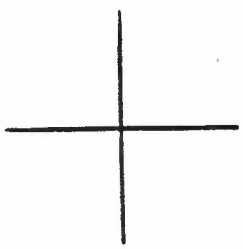
Colour Index points to same colour description as polyline

MARKER TYPE

< 0	implementation dependent	
0	invalid	
1	dot	•
2	plus	+
3	star	*
4	circle	o
5	cross	×
⩾6	subject to registration	

MARKER SIZE SCALE FACTOR

3·0



1·0 standard

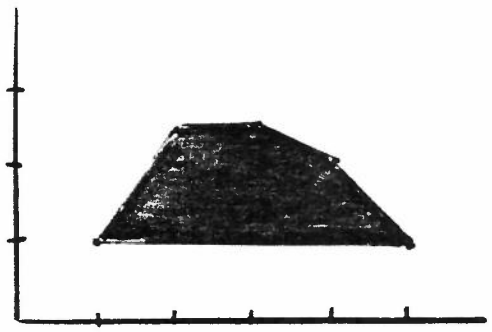
0·5

FILL AREA (n , list of points)

CALL GFA (N, XA , YA)

The set of points define a boundary of an area to be filled in with a pattern

FILL AREA EXAMPLE



FILL AREA (6,XA,YA)

N	XA	YA
1	1	1
2	2	2·5
3	3	2·5
4	4	2
5	5	1
6	1	1

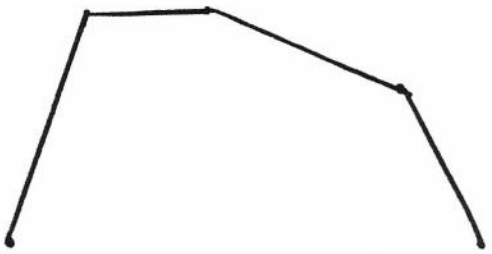
BOUNDARY CLOSURE

If last point is not same as first, additional point is added equal to first.

```
FILL AREA (5, XA, YA)
FILL AREA (6, XA, YA)
```

in the previous example produce same output

POLYLINE (S, XA, YA)



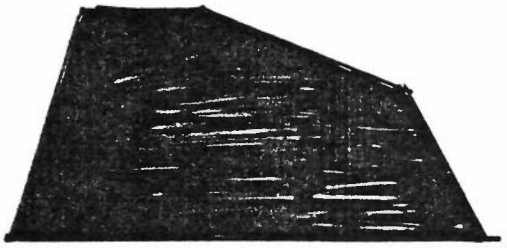
POLYMARKER (S, XA, YA)

* *

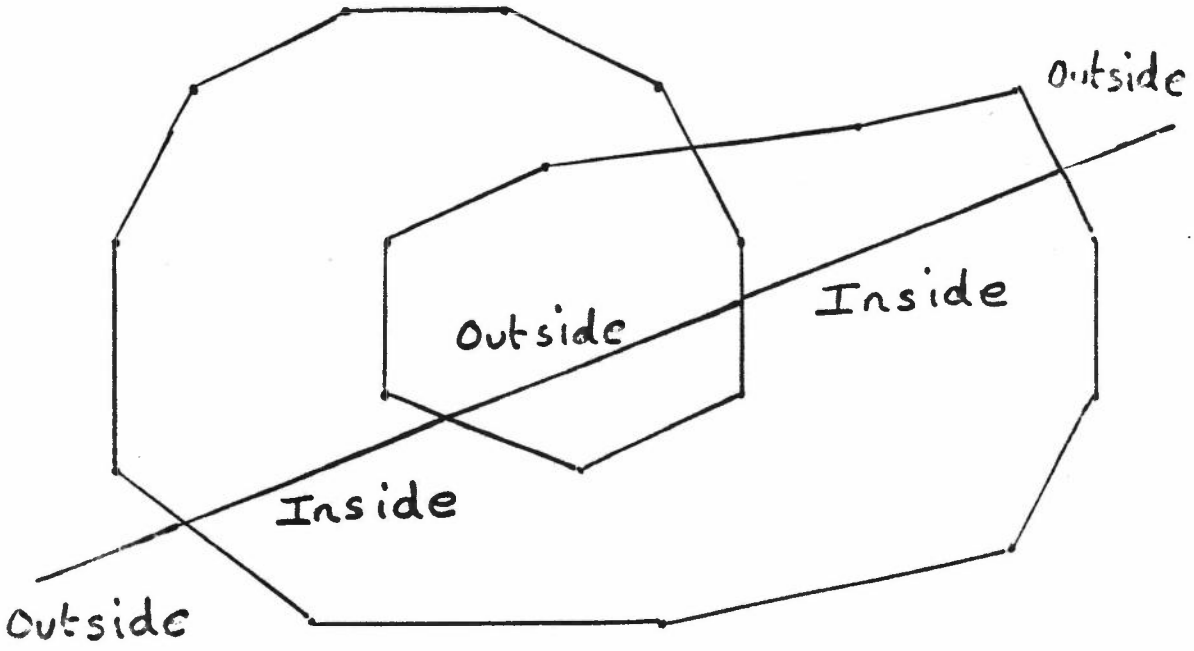
*

* *

FILL AREA (S, XA, YA)

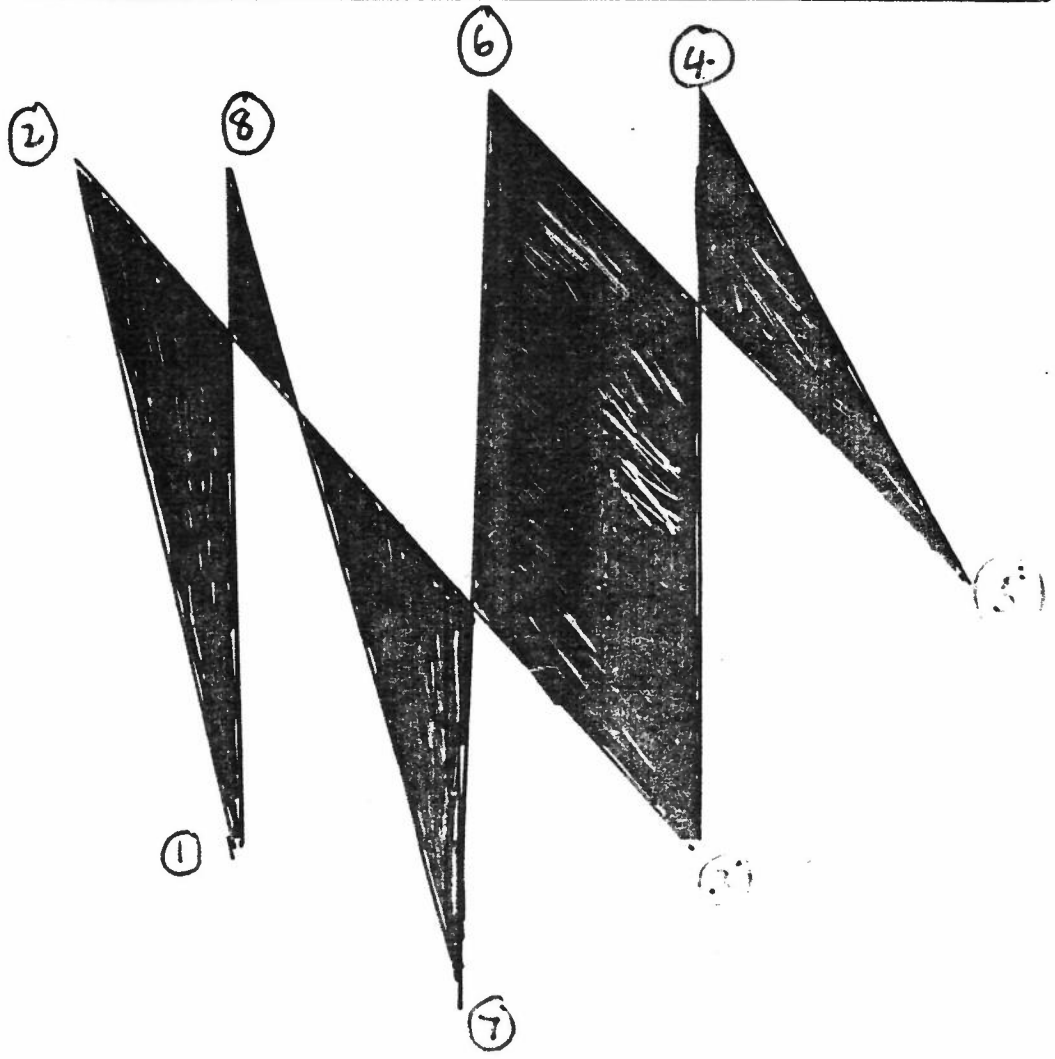


BOUNDARY CROSSING OVER ITSELF



Point is inside boundary if arbitrary point at infinity has a line from it which intersects the boundary an odd number of times

EXAMPLE OF CROSSING BOUNDARY



FILL AREA ASPECTS

The set of points define the boundary of the fill area

HOW the area is filled is controlled by the aspects of the fill area

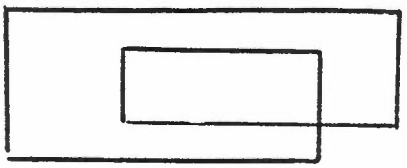
- Interior Style
- Style Index
- Colour Index
- Pattern Reference Point
- Pattern Size

Colour Index points to same colour descriptions as polyline and polymarker

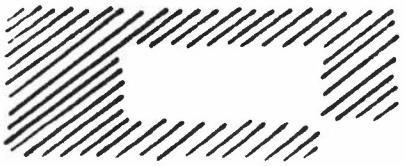
INTERIOR STYLE

Solid

Hollow



Hatch



Pattern

STYLE INDEX

Selects specific hatch or pattern style

HATCH

PATTERN

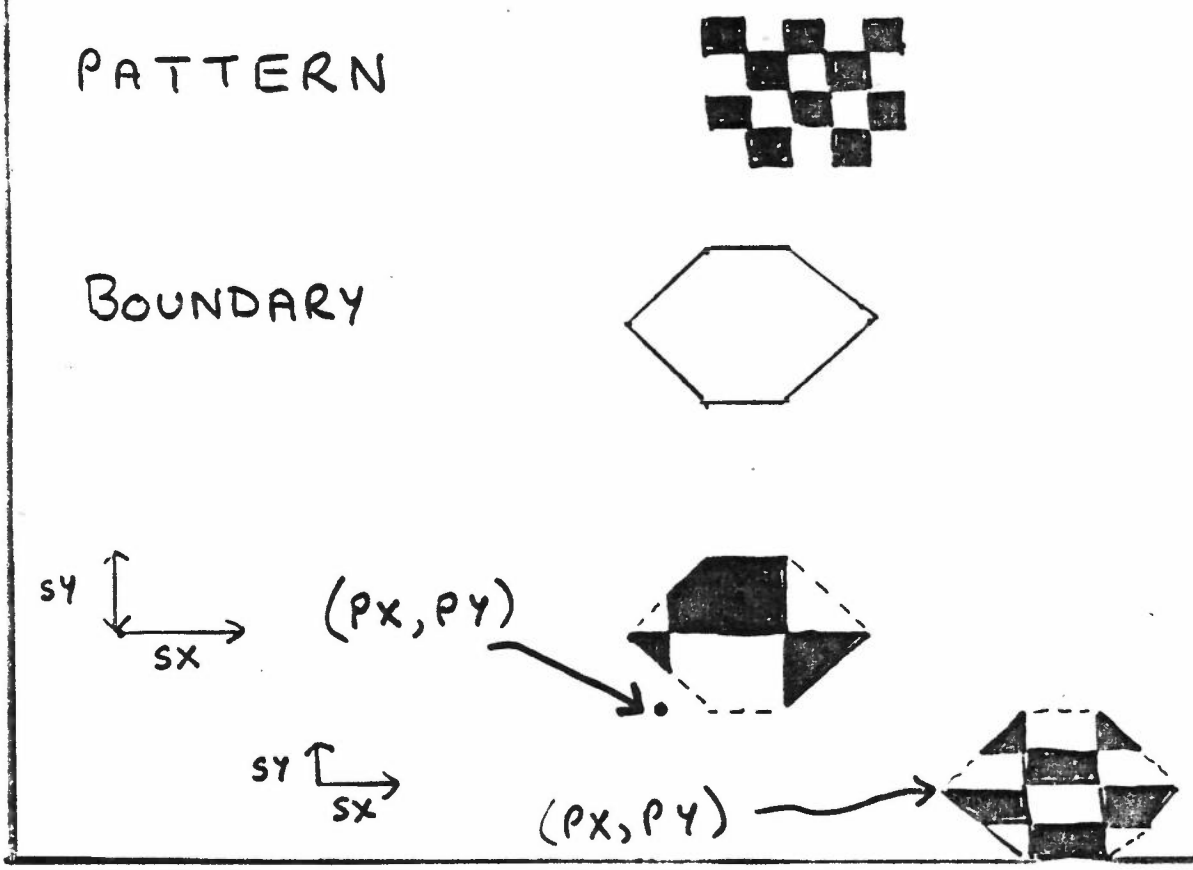
PATTERN SIZE and PATTERN REFERENCE POINT

The size and origin of the pattern can be defined by

SET PATTERN SIZE (SX, SY)

and

SET PATTERN REFERENCE POINT (PX, PY)



TEXT (starting point, character string)

CALL GTX (X, Y, CHA)

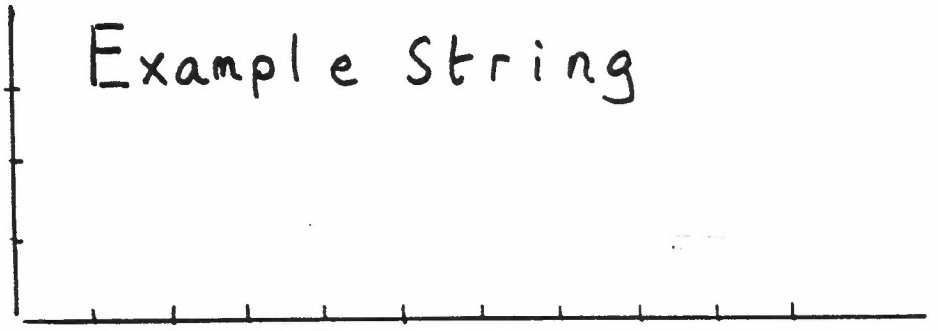
The character string specified is output at the starting point given

TEXT EXAMPLE

CHA

E	x	a	m	p	l	e		S	t	r	i	n	g

TEXT (1, 3, CHA)



TEXT ASPECTS

The origin of the text string and the characters in the string are specified by the TEXT primitive

HOW the characters are formed, their size and orientation are controlled by the aspects of the text

- Character Height
- Character Up Vector
- Colour Index
- Text Path
- Text Alignment
- Text Font and Precision
- Character Expansion Factor
- Character Spacing

Colour Index points to same colour descriptions as other primitives

CHARACTER HEIGHT

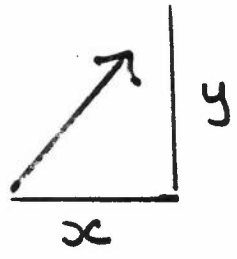
3.0 Height

2.0 Height

1.0 Height

0.5 Height

CHARACTER UP VECTOR



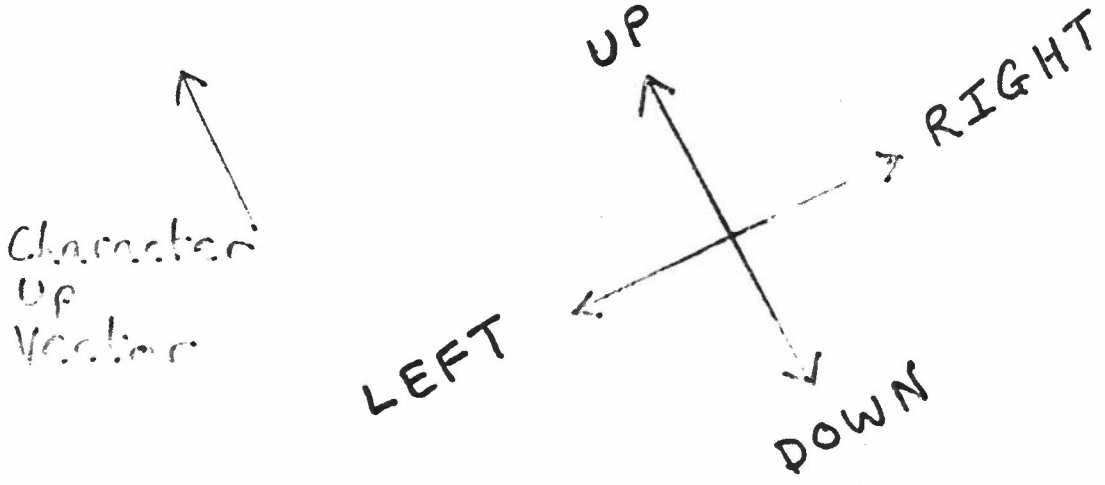
0,1 ABC

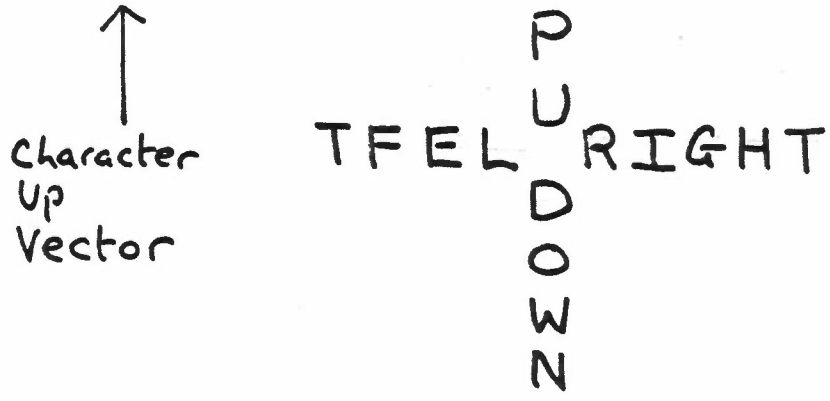
-1,1 ABC

1,0 ABC

TEXT PATH

Writing direction for different values of text path





TEXT ALIGNMENT

.Left Aligned

Right Aligned.

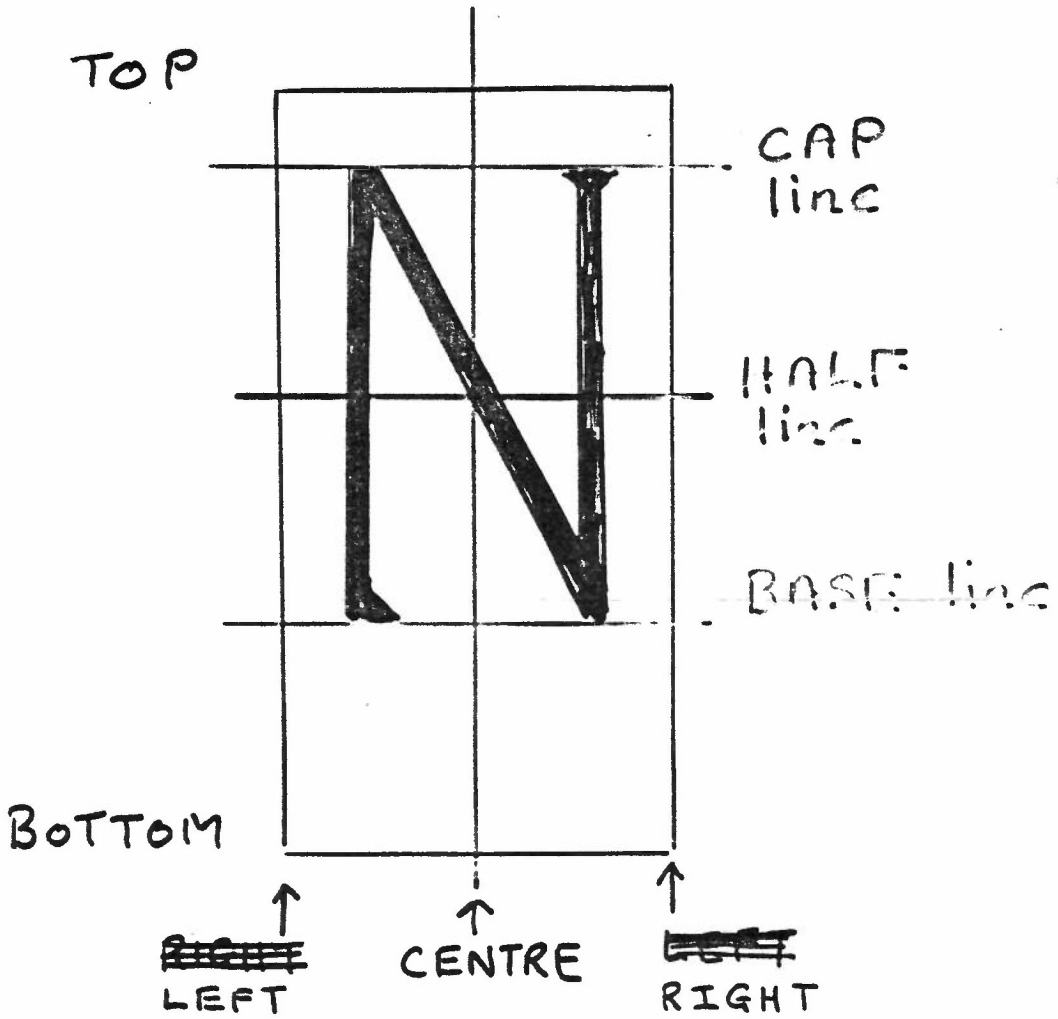
Centre. Aligned

•TOP .BASE

.Bottom Aligned

•CAP •HALF

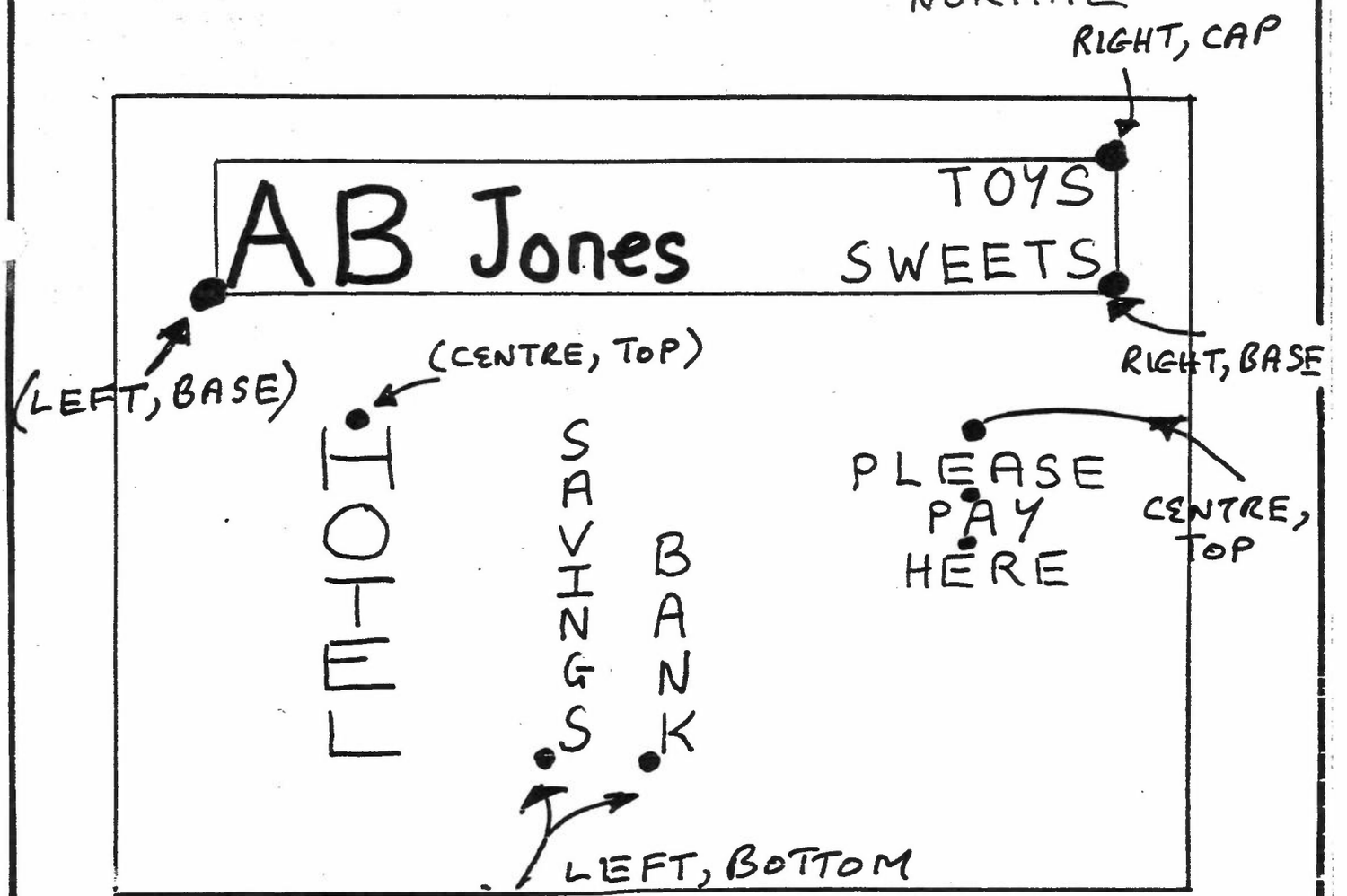
FONT DESIGN



Font Designer specifies shape of each character. Fonts can be monospaced or proportional. Width of character includes space on each side.

Text may be aligned separately along HORIZONTAL and VERTICAL axes.

HORIZONTAL	VERTICAL
LEFT	TOP
CENTRE	CAP
RIGHT	HALF
NORMAL	BASE
	BOTTOM
	NORMAL



NORMAL ALIGNMENT

For each value of TEXT PATH there is an alignment which is NORMAL and is equivalent to one of the other options

TEXT PATH	NORMAL Alignment	
	HORIZONTAL	VERTICAL
RIGHT	LEFT	BASE
LEFT	RIGHT	BASE
UP	CENTRE	BASE
DOWN	CENTRE	TOP

TEXT FONT AND PRECISION

Defines which font to be used and the precision with which the text aspects are executed

STRING	only CHARACTER HEIGHT needs to be executed
CHAR	character boxes should be accurately aligned
STROKE	all aspects executed

CHARACTER EXPANSION FACTOR

Expansion 1·0

Expansion 0·5

nsion 1·5

7 3·0

CHARACTER SPACING

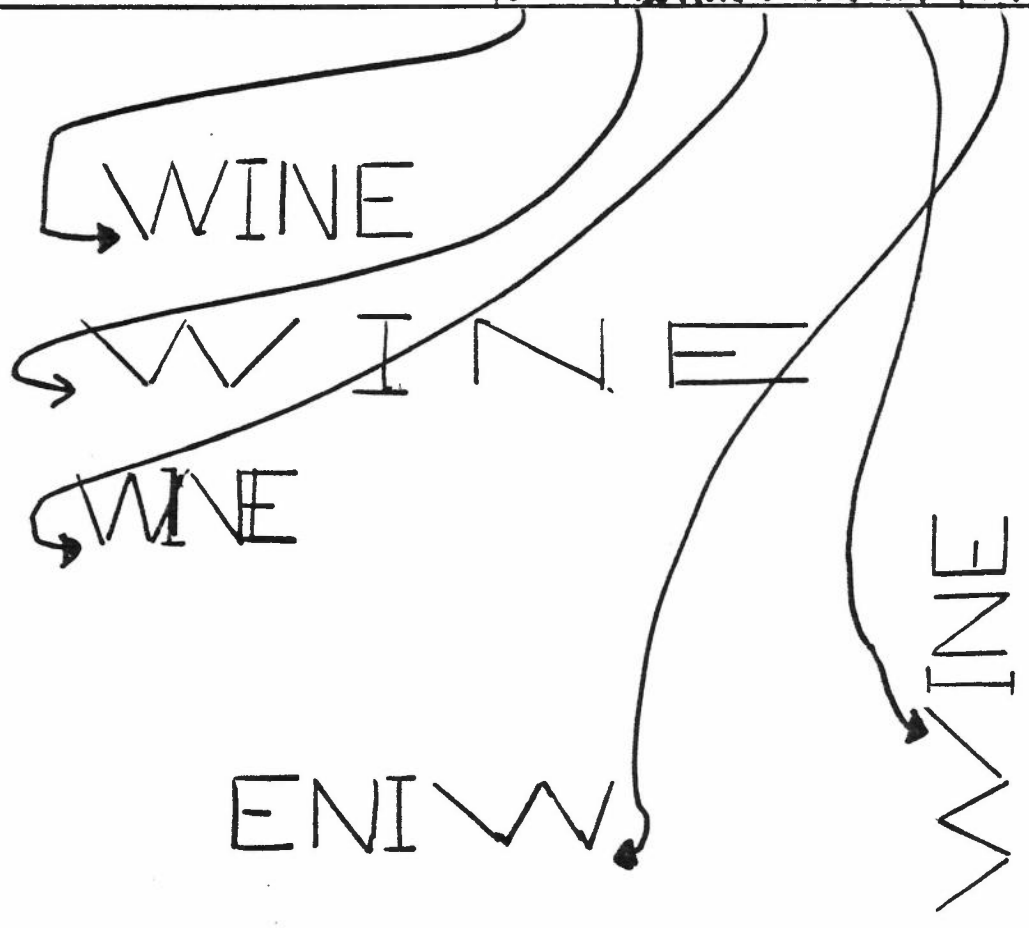
Spacing 0

acing 0·3

ng 1·0

Character spacing is defined as a fraction of CHARACTER HEIGHT

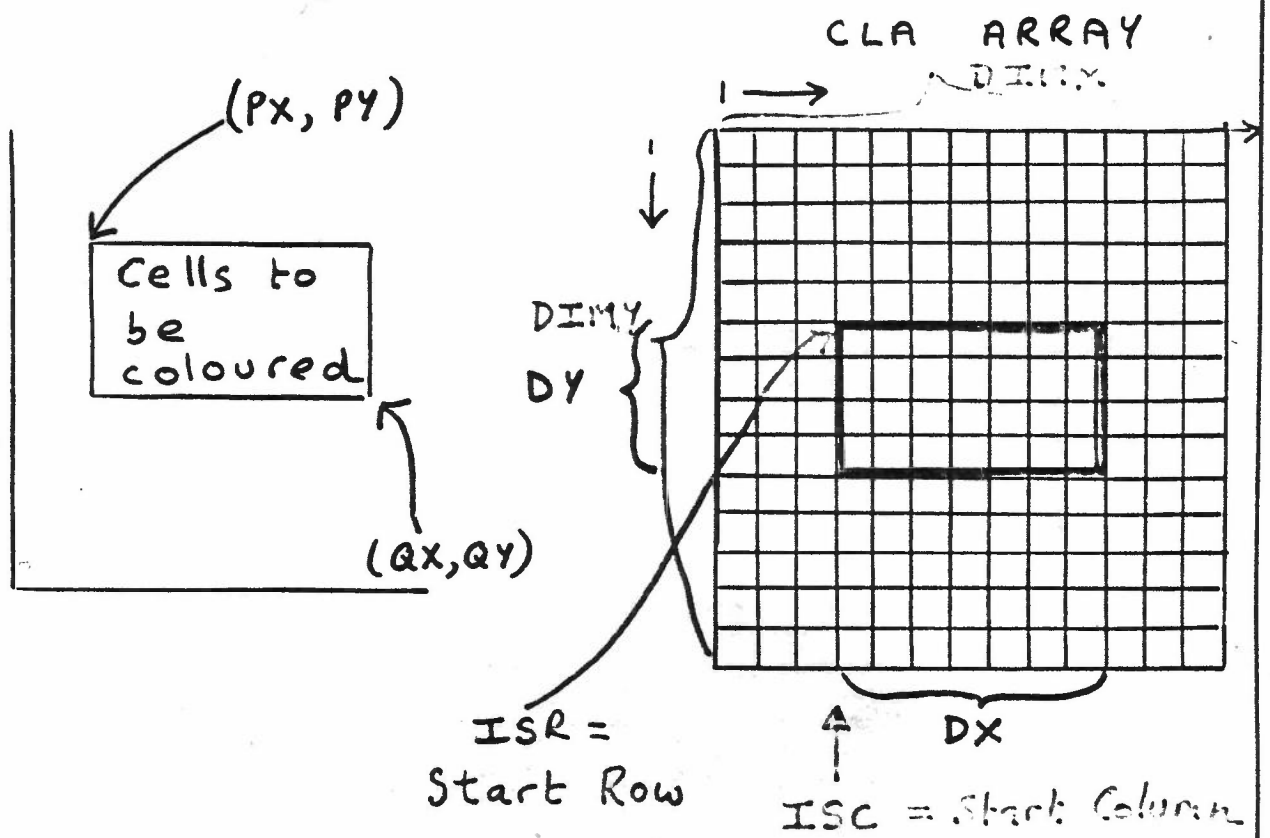
CHARACTER HEIGHT	1	1	1	1	1
CHARACTER EXPANSION FACTOR	1	2	1	1	1
CHARACTER SPACING	0	0.5	-0.5	0	0
CHARACTER UP VECTOR	0,1	0,1	0,1	-1,0	0,1
TEXT PATH	RIGHT	RIGHT	RIGHT	RIGHT	LEFT
TEXT ALIGNMENT	LEFT BOTTOM	LEFT BOTTOM	LEFT BOTTOM	CENTRE TOP	RIGHT BOTTOM



CELL ARRAY

CALL GCA (PX, PY, QX, QY, DIMX , DIMY, ISC, ISR, DX, DY, CLA)

The array of cells specified are coloured by the indices given in CLA



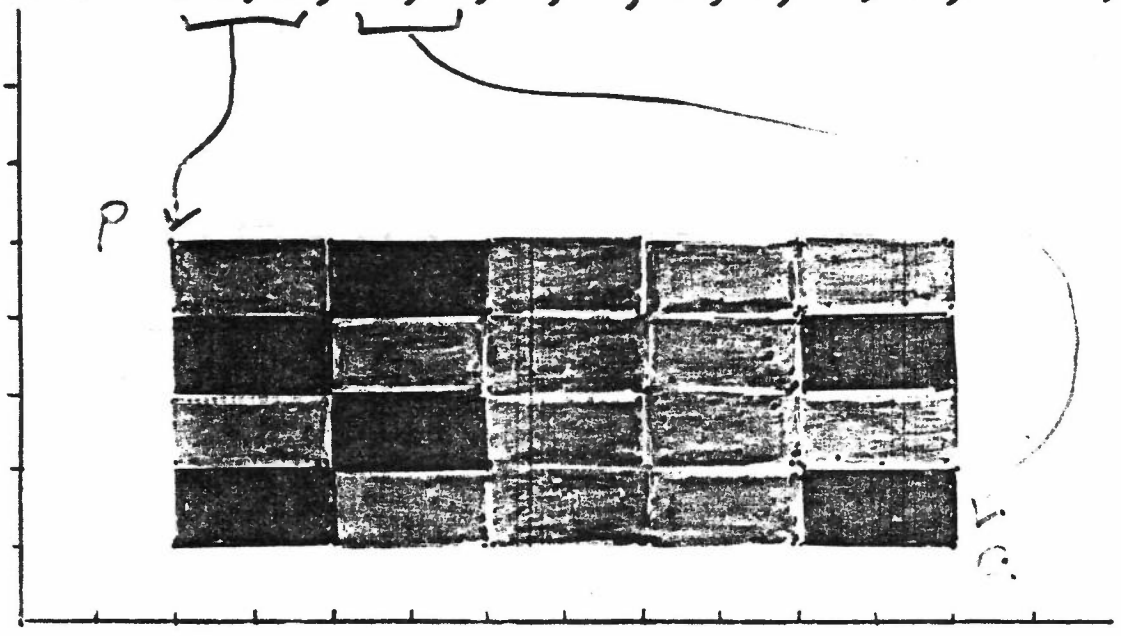
Point P is at cell coloured by CLA (ISC, ISR)

Point Q is at cell coloured by CLA (ISC + DX - 1, ISR + DY - 1)

CLA

	1	2	3	4	5	6	7
1							
2							
3		1	2	1	?	1	
4		2	1	?	1	2	
5		1	2	1	?	1	
6		2	1	?	1	2	
7							
8							

CALL GCA(2, 5, 12, 1, 7, 8, 2, 3, 5, 4, CLA)



COORDINATE SYSTEMS

USER — The application may require the problem to be defined in a specific coordinate system

WORLD — GKS requires all problems to be defined in cartesian coordinate systems

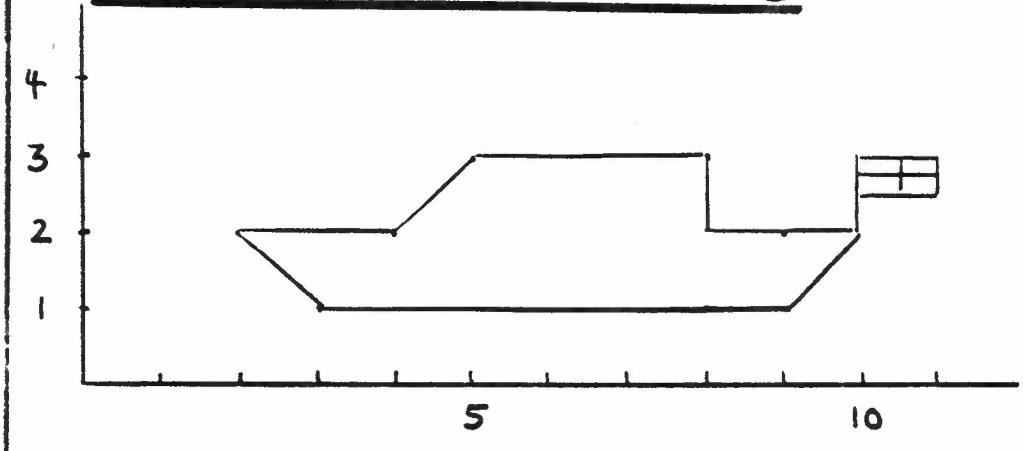
NORMALIZED DEVICE — GKS defines a virtual drawing area extending from [0,1] in the X and Y directions on to which all GKS output in world coordinates is mapped

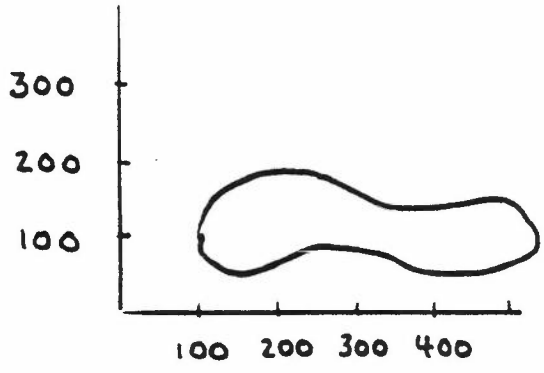
DEVICE

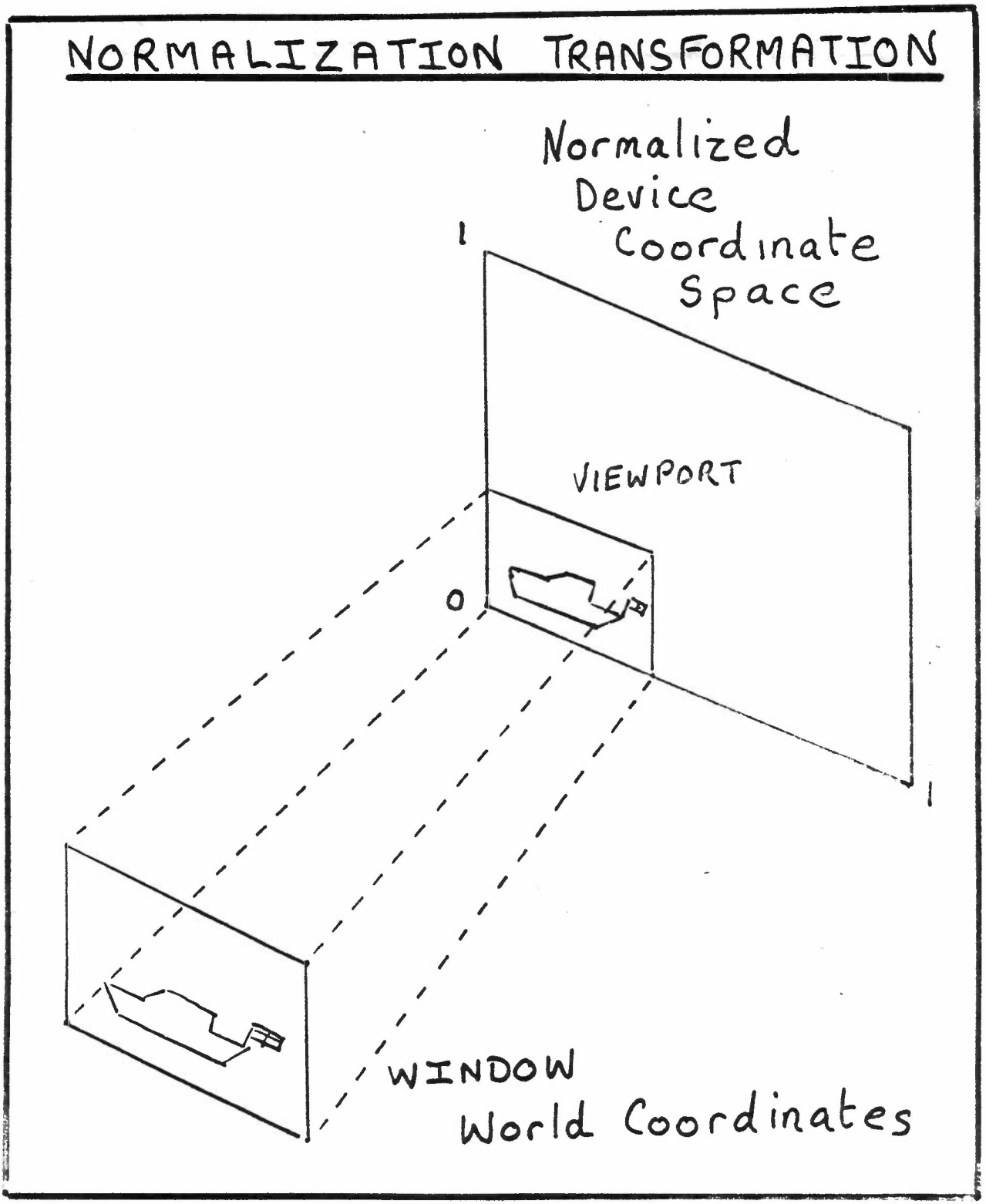
NORMALIZATION TRANSFORMATION

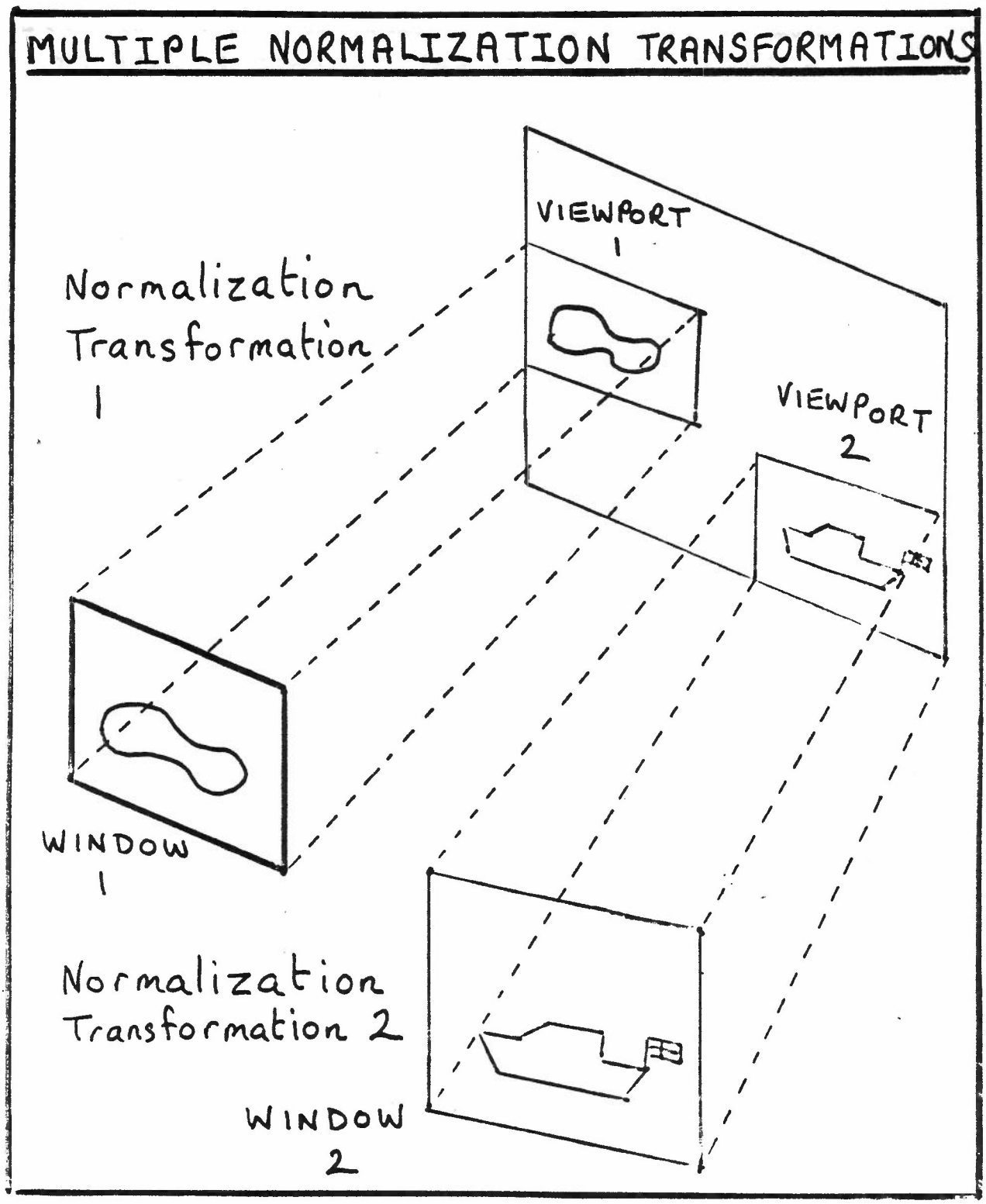
A window in world coordinate space is mapped to a viewport in the virtual drawing area : coordinates are normalized device coordinates

WORLD COORDINATES





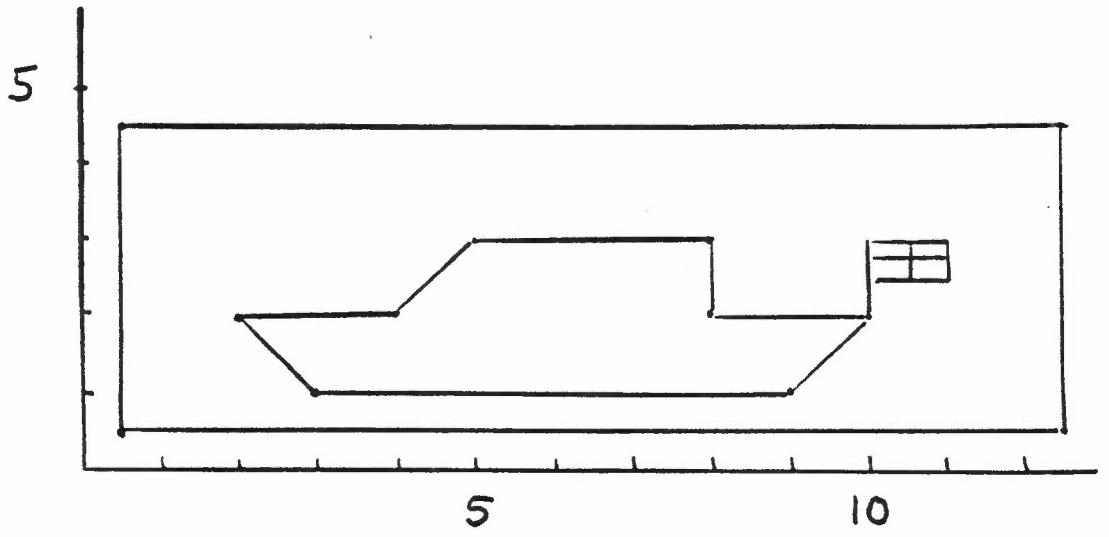

NORMALIZATION TRANSFORMATION
Normalized
Device
Coordinate
Space
1
VIEWPORT
0
1
WINDOW
World Coordinates


MULTIPLE NORMALIZATION TRANSFORMATIONS
VIEWPORT 1
Normalization Transformation 1
VIEWPORT 2
WINDOW 1
Normalization Transformation 2
WINDOW 2

SET WINDOW (TN, window limits)

CALL GSWN(TN,XMN,XMX,YMN,YMX)

Defines the window of normalization transformation TN

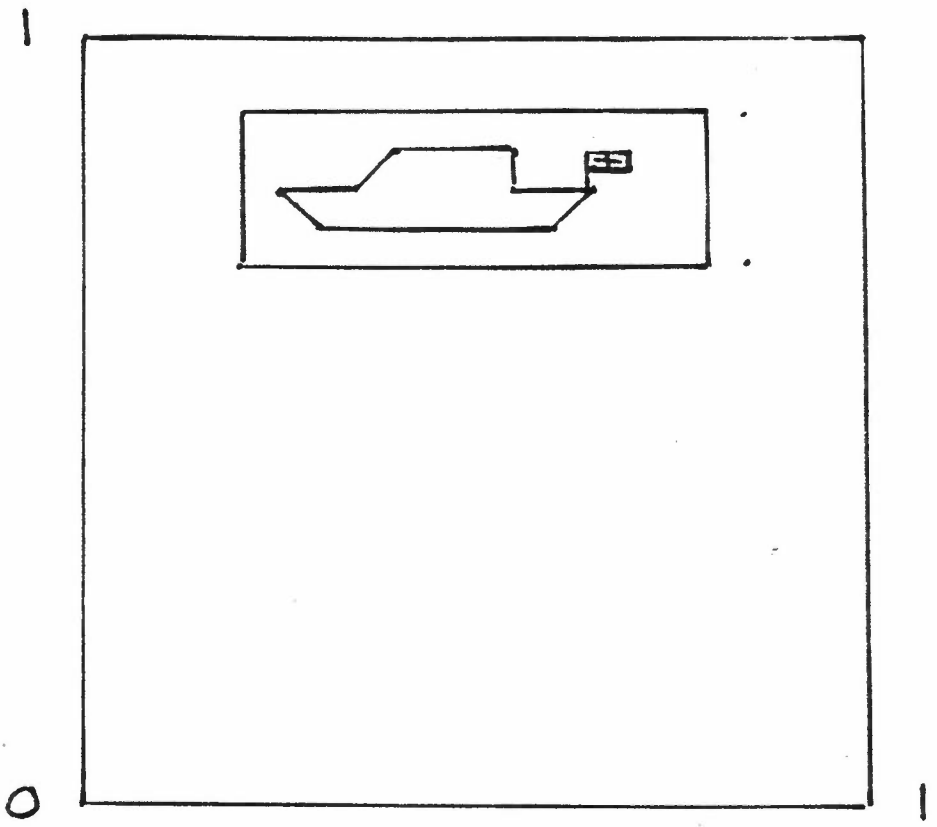


GSWN (TN, 0·5, 12·5, 0·5, 4·5)

SET VIEWPORT (TN , viewport limits)

CALL GSVP (TN , XVMN , XVMX , YVMN , YVMX)

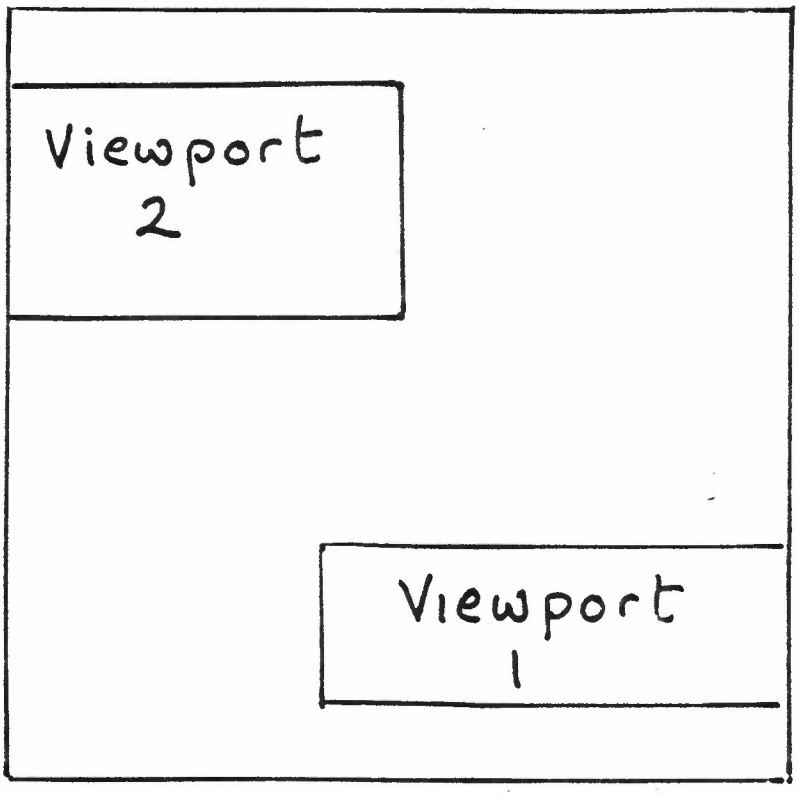
Defines the viewport of normalization transformation TN



GSVP (TN , 0·2 , 0·8 , 0·7 , 0·9)

MULTIPLE NORMALIZATION TRANSFORMATIONS

```
SET WINDOW (1, 0·5, 12·5, 0·5, 4·5)
SET VIEWPORT(1, 0·4, 1·0, 0·1, 0·3)
SET WINDOW (2, 0, 500, 0, 300)
SET VIEWPORT(2, 0, 0·5, 0·6, 0·9)
```



SELECT NORMALIZATION TRANSFORMATION (TN)

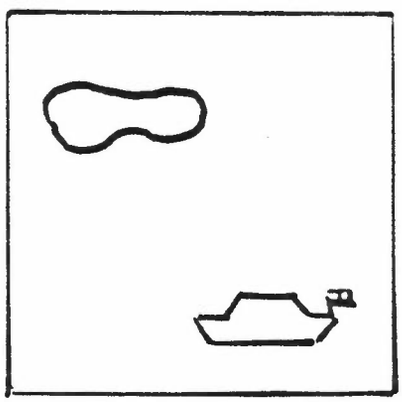
CALL GSELNT(TN)

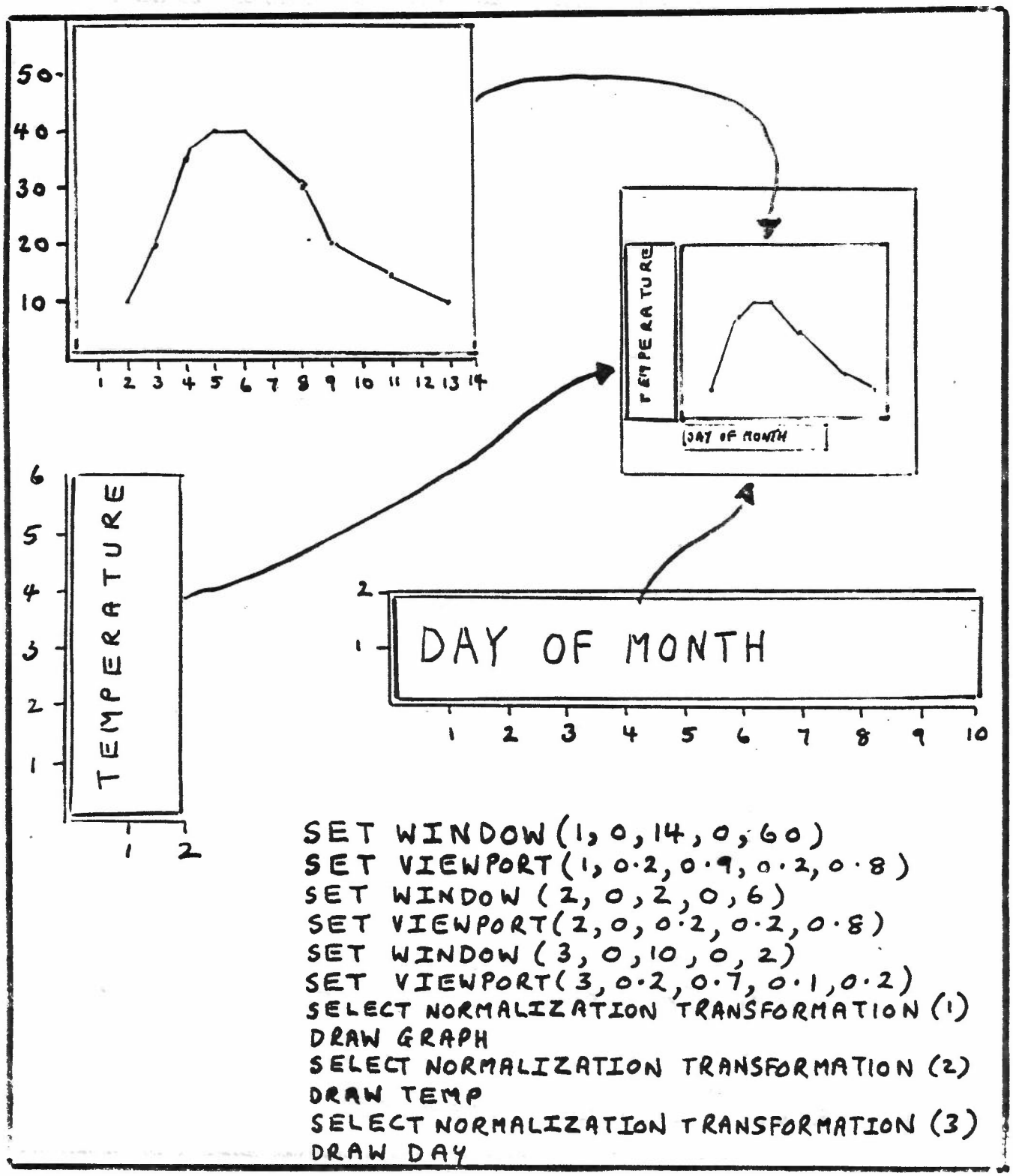
Selects the normalization transformation to be used by subsequent output primitives until another normalization transformation is called

EXAMPLE

SELECT NORMALIZATION TRANSFORMATION (1)
BOAT

SELECT NORMALIZATION TRANSFORMATION(2)
ISLAND






```
SET WINDOW (1, 0, 14, 0, 60)
SET VIEWPORT (1, 0·2, 0·9, 0·2, 0·8)
SET WINDOW (2, 0, 2, 0, 6)
SET VIEWPORT (2, 0, 0·2, 0·2, 0·8)
SET WINDOW (3, 0, 10, 0, 2)
SET VIEWPORT (3, 0·2, 0·7, 0·1, 0·2)
SELECT NORMALIZATION TRANSFORMATION (1)
DRAW GRAPH
SELECT NORMALIZATION TRANSFORMATION (2)
DRAW TEMP
SELECT NORMALIZATION TRANSFORMATION (3)
DRAW DAY
```

MULTIPLE NORMALIZATION TRANSFORMATIONS

- Multiple Normalization Transformations allow all transformations to be active that make up the picture

A CONVENTIONAL PACKAGE

```
SET WINDOW (0,14,0,60)
SET VIEWPORT (0·2,0·9,0·2,0·8)
DRAW GRAPH
SET WINDOW (0,2,0,6)
SET VIEWPORT(0,0·2,0·2,0·8)
DRAW TEMP
SET WINDOW (0,10,0,2)
SET VIEWPORT(0·2,0·7,0·1,0·2)
DRAW DAY
```

At this point all knowledge of other two coordinate systems used earlier has gone

For locator input, only possibility is to return Device Coordinates

NORMALIZATION TRANSFORMATIONS

GOOD PRACTICE

- Define all normalization transformations at head of program

  Analagous to variable declarations

- Keep all transformations known to system means user's model of system is close to reality

NORMALIZATION TRANSFORMATION 0

Normalization transformation 0 cannot be changed. Set to:

```
SET WINDOW (0, 0, 1, 0, 1)
SET VIEWPORT (0, 0, 1, 0, 1)
```

Output specified with respect to Normalization Transformation 0 effectively uses NDC coordinates

ASPECT RATIO

Normalization Transformations do not need to preserve aspect ratio

Normalization Transformations define the coordinate transformation from world to NDC

WINDOW → VIEWPORT

SET WINDOW (1, 0·5, 12·5, 0·5, 4·5)
SET VIEWPORT (1, 0·0, 0·6, 0·0, 0·4)

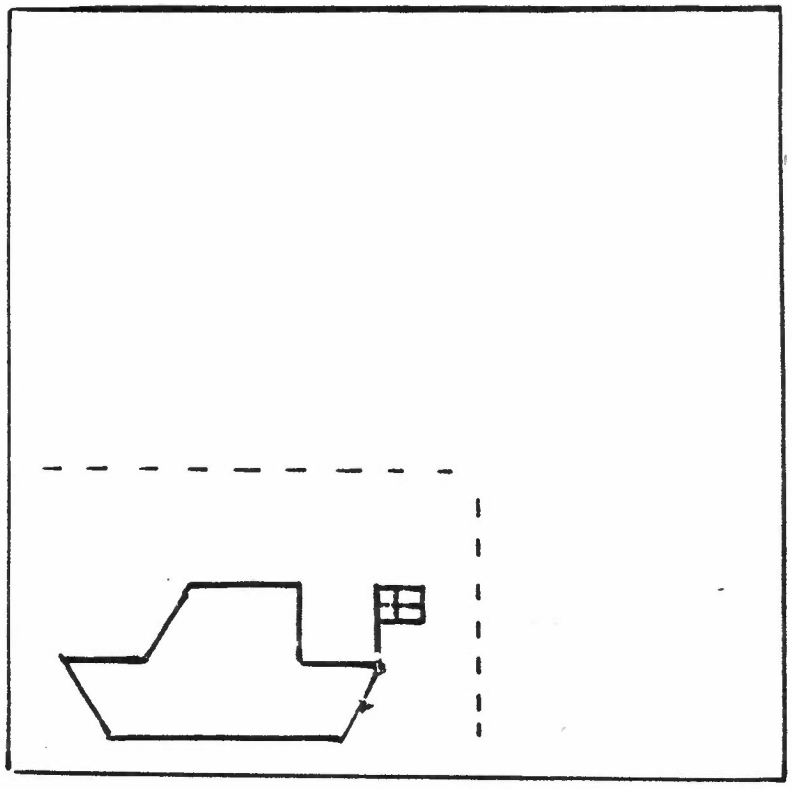
is same as :

SET WINDOW (1, 3·5, 9·5, 1·5, 3·5)
SET VIEWPORT (1, 0·15, 0·45, 0·1, 0·3)

ASPECT RATIO

SET WINDOW (1, 0·5, 12·5, 0·5, 4·5)

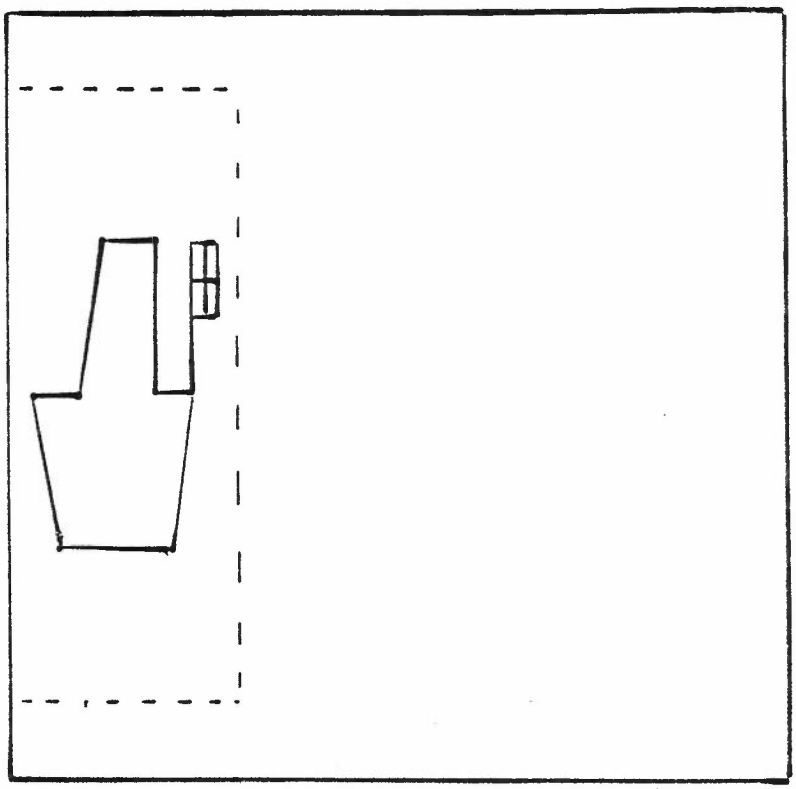
SET VIEWPORT (1, 0·0, 0·6, 0·0, 0·4)



ASPECT RATIO

SET WINDOW(1, 0·5, 12·5, 0·5, 4·5)

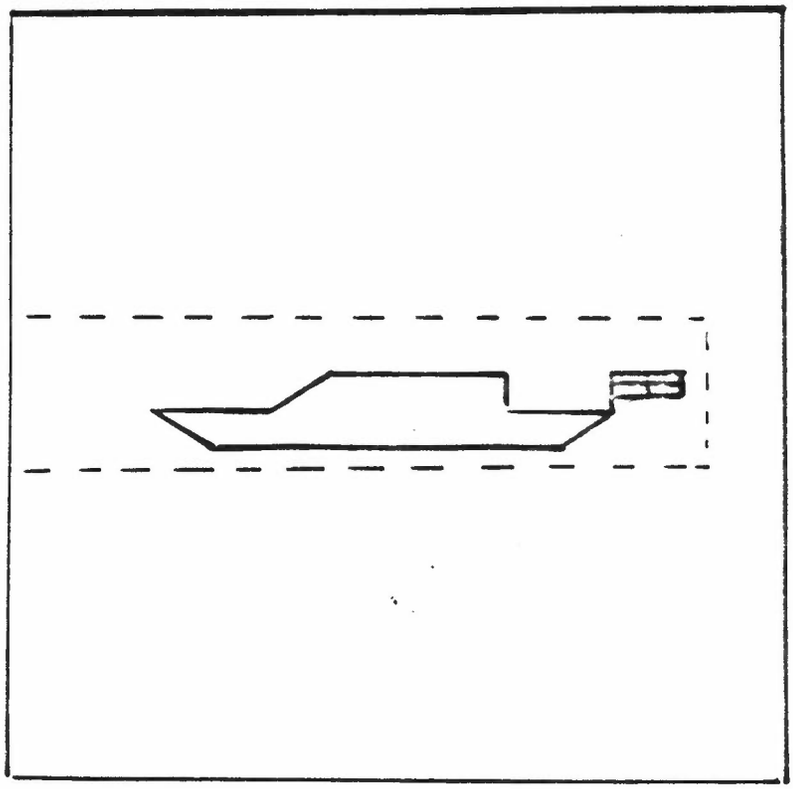
SET VIEWPORT(1, 0·0, 0·3, 0·1, 0·9)



ASPECT RATIO

SET WINDOW (1, 0·5, 12·5, 0·5, 4·5)

SET VIEWPORT(1, 0·0, 0·9, 0·4, 0·6)



CLIPPING

The boundaries of window and viewport primarily define the coordinate transformation from world to NDC coordinates

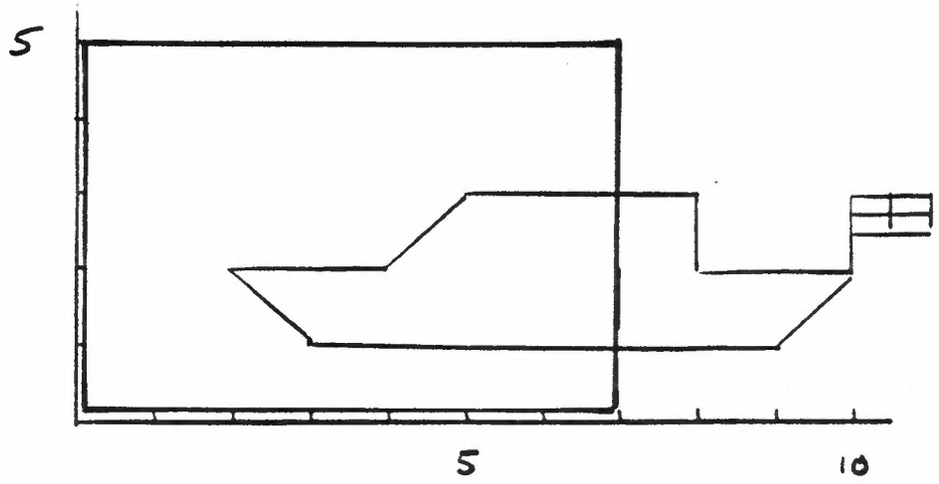
The boundary can also be used to clip output primitives so that only those parts of primitives inside the boundary are drawn.

SET CLIPPING INDICATOR (CLIP)

CALL GSCLIP (CLSW)

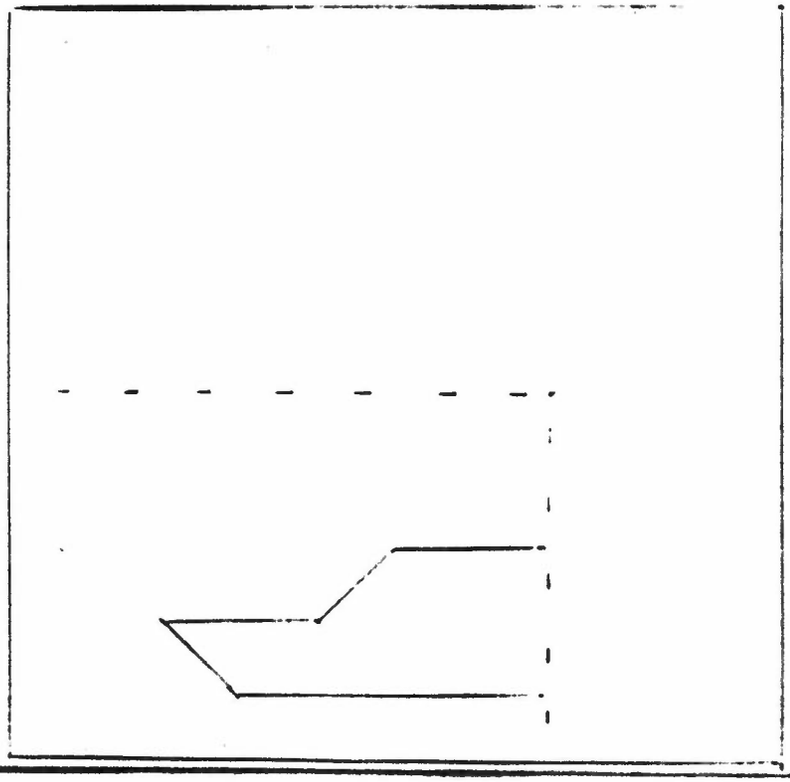
Clipping is to the window of the currently selected normalization transformation

CLIPPING



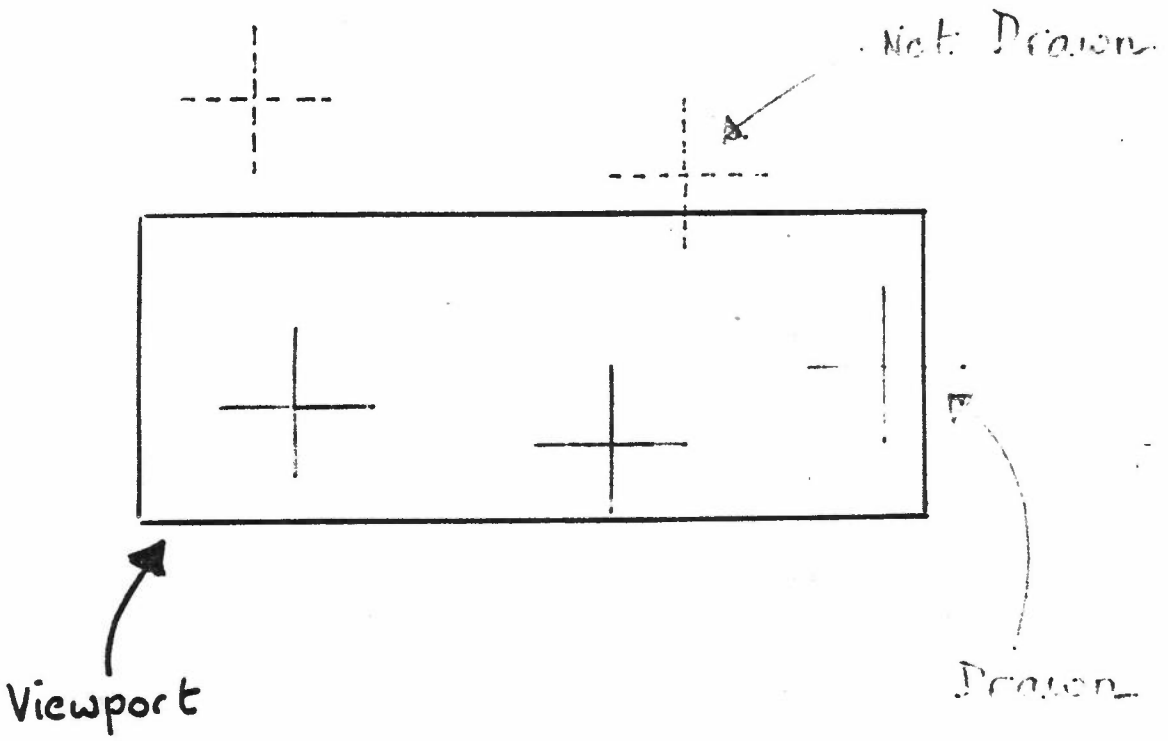
SET WINDOW (1, 0, 7, 0, 5)

SET VIEWPORT (1, 0·0, 0·7, 0·0, 0·5)

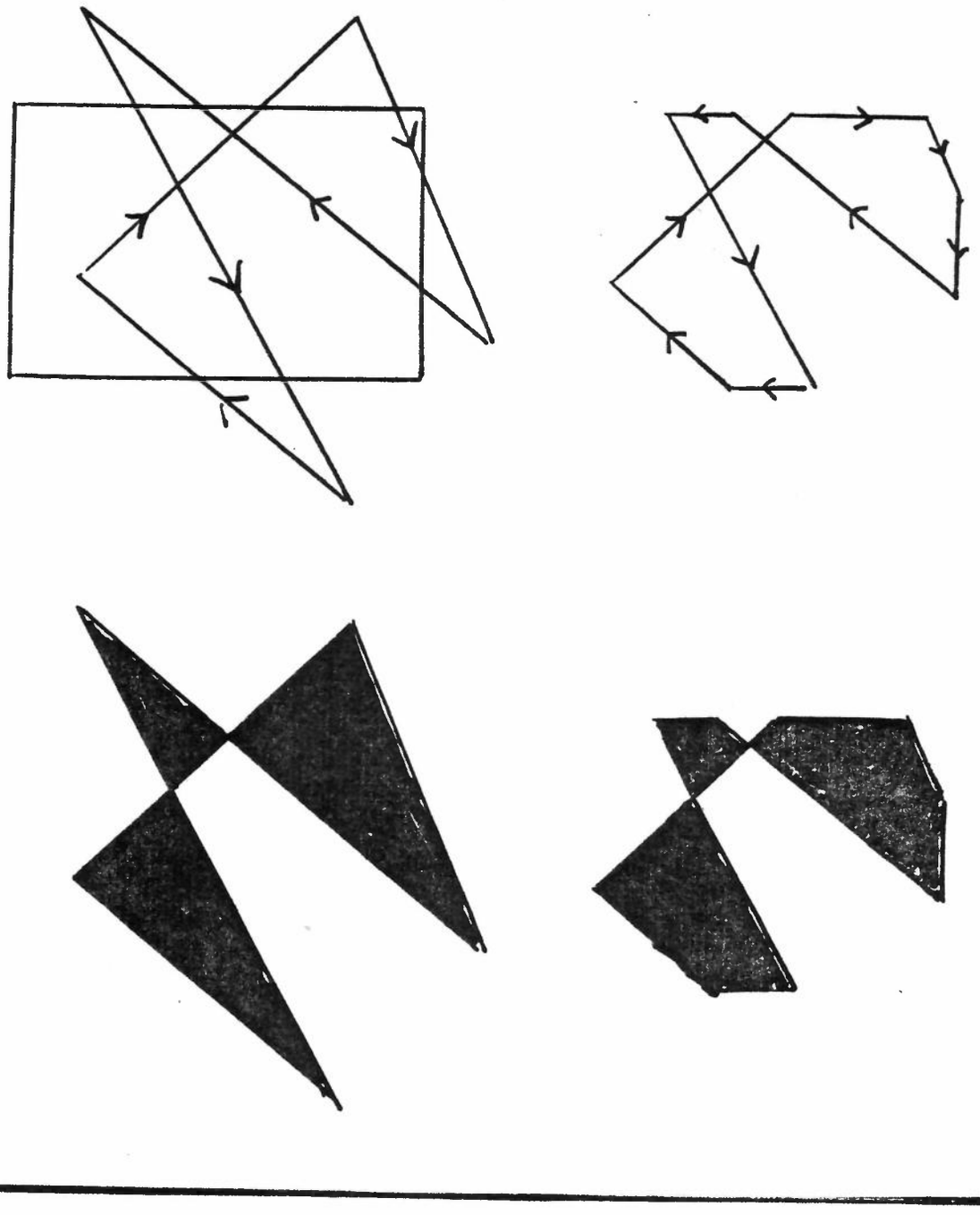


CLIPPING POLYMARKER

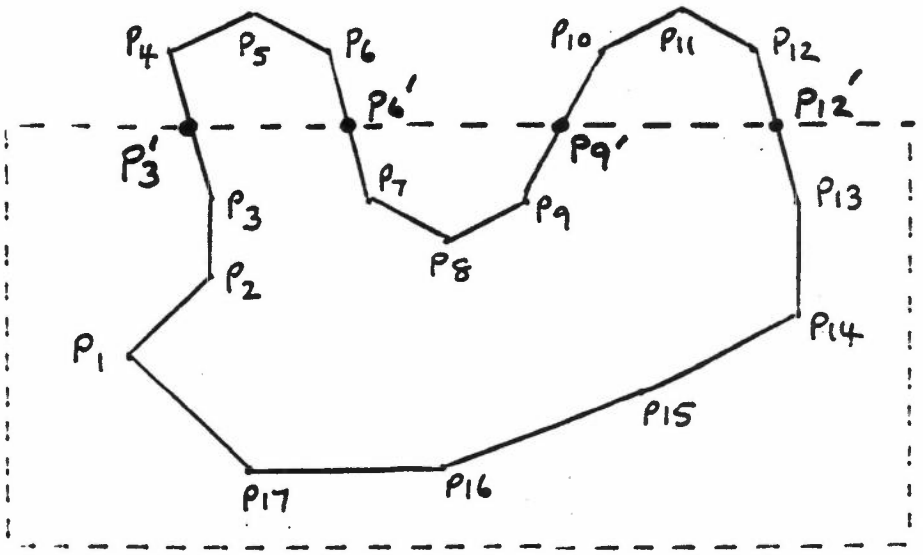
Marker is output if centre of marker is within viewport



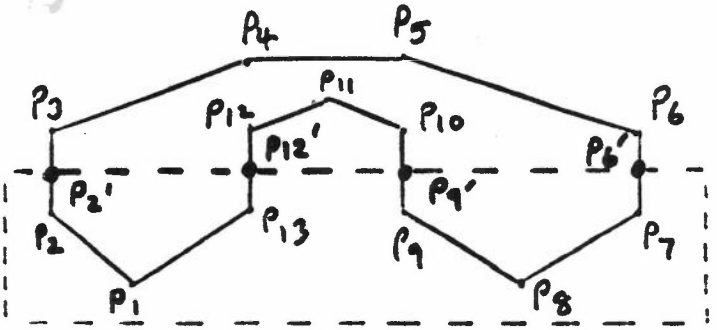
CLIPPING FILL AREA



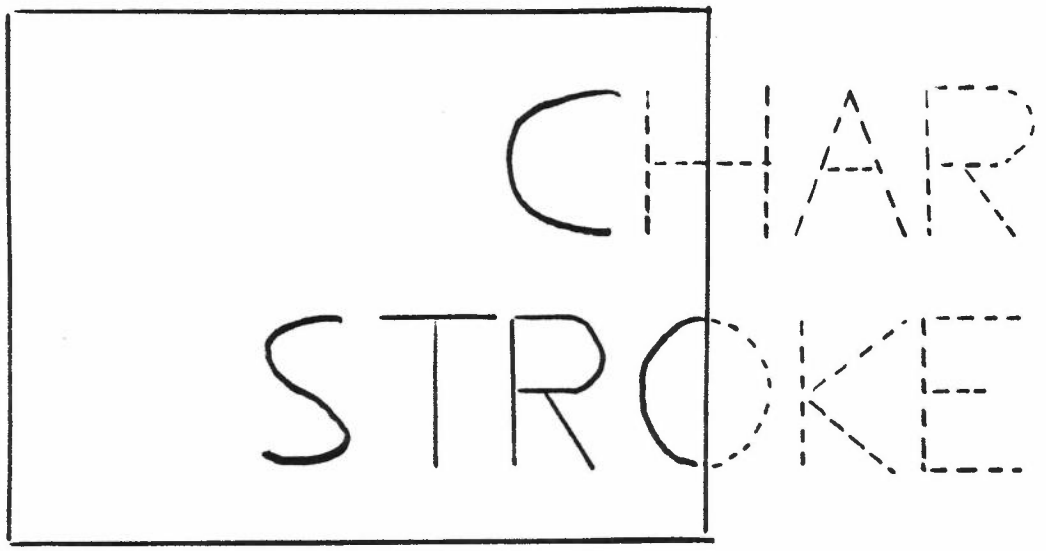
CLIPPING A FILL AREA PRIMITIVE



- Clipping a fill area primitive causes additional points P_3' P_6' P_9' P_{12}'
- Polyon filled is $P_1 \rightarrow P_3\ P_3'\ P_6'\ P_7 \rightarrow P_9\ P_9'\ P_{12}'\ P_{13}$ ..
- Clipping a fill area polygon may produce several subareas



CLIPPING TEXT



STRING text : implementation dependent

ATTRIBUTES

- Appearance of an output primitive on a workstation is controlled by its attributes

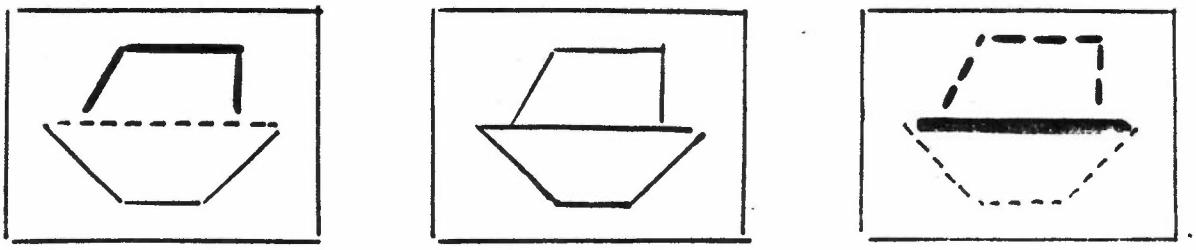
- An attribute of a primitive consists of one or more aspects

GKS uses attributes to:

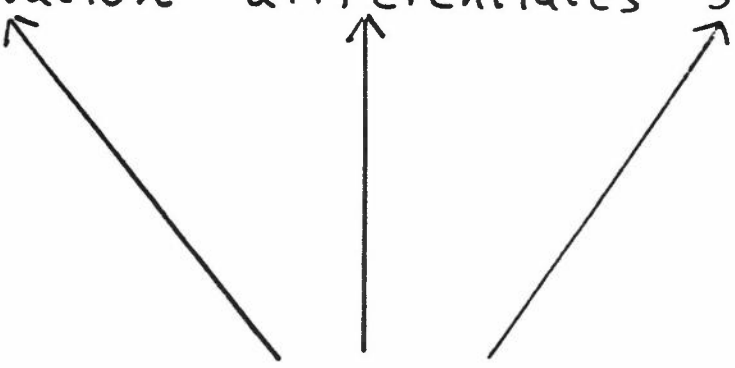
- Ensure that primitives with different attributes are differentiable on every workstation if possible — BUNDLED MODE
- Ensure that primitives with specific attributes are represented on workstations as close as possible to the specification - UNBUNDLED

BUNDLED ATTRIBUTES

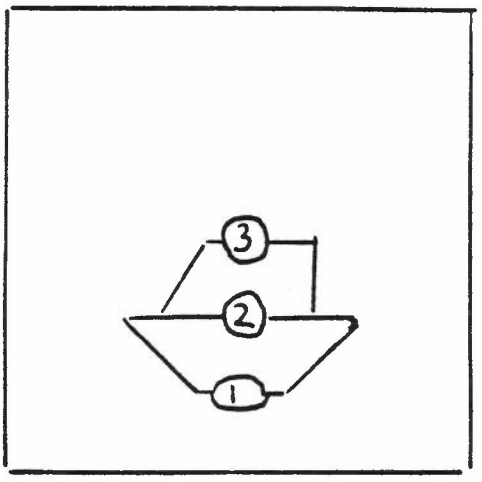
WORKSTATIONS



Workstation differentiates somehow

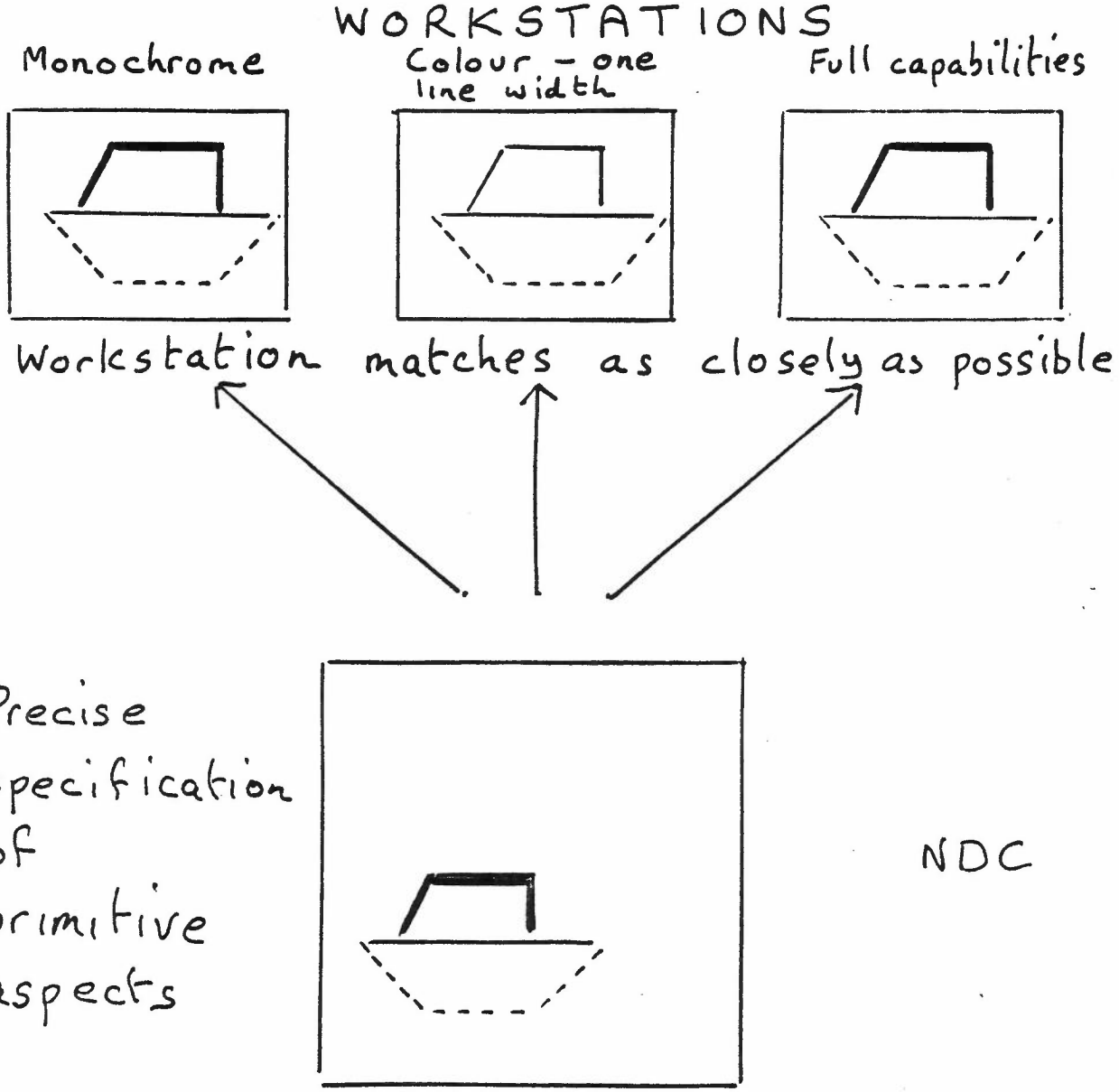


Primitives marked as having different attributes



NDC

UNBUNDLED ATTRIBUTES



GKS WORKSTATION

- User environment consists of one or more workstations
- GKS workstation has a single display surface
- Application programs should be movable between different user environments without major changes to program structure
- Workstations are responsible for defining how primitive attributes are mapped on to real devices

ATTRIBUTES

Aspects of primitives control the appearance of primitives

GEOMETRIC ASPECTS

control shape or size of primitive

NON-GEOMETRIC ASPECTS

control appearance, not shape or size

GEOMETRIC ASPECTS

Each is controlled by a single geometric attribute

Only output primitives having geometric attributes are:

TEXT : Character Height
Character Up Vector
Text Path
Text Alignment

FILL AREA: Pattern Size
Pattern Reference Point

PRIMITIVE	ASPECTS	
	WORKSTATION DEFENDENT	GLOBAL
POLYLINE	Linetype Linewidth Scale Factor Polyline Colour Index	None
POLYMARKER	Markertype Marker size scale factor Polymarker Colour Index	None
FILL AREA	Fill area interior style Fill area style index Fill area Colour Index	Pattern reference point Pattern size
TEXT	Text font and precision Character expansion factor Character spacing Text Colour Index	Character height Character up vector Text path Text alignment
CELL ARRAY	None	None

GEOMETRIC ATTRIBUTES

- Workstation Independent
- Specified in World Coordinates
- Subject to Normalization and Workstation Transformation
- Set Modally
- Bound to Output Primitive when it is defined

ATTRIBUTE SETTING FUNCTIONS

SET CHARACTER HEIGHT
CALL GSCHH (HEIGHT)

SET CHARACTER UP VECTOR
CALL GSCHUP (CHUPX, CHUPY)

SET TEXT PATH
CALL GSTXP (TXP)

SET TEXT ALIGNMENT
CALL GSTXAL(TXALH, TXALV)

SET PATTERN SIZE
CALL GSPA (SZX, SZY)

SET PATTERN REFERENCE POINT
CALL GSPARF(RFX, RFY)

NON-GEOMETRIC ASPECTS

Controlled in one of two ways:

BUNDLED

One attribute per output primitive controls all non-geometric aspects

Attributes are: POLYLINE INDEX
POLYMARKER INDEX
TEXT INDEX
. . . . . .

INDIVIDUAL

One attribute for each non-geometric aspect

POLYLINE INDEX

POLYLINE INDEX is index into Polyline Bundle Table which contains values of all the non-geometric aspects

A separate Bundle Table exists for each workstation

POLYLINE BUNDLE TABLE

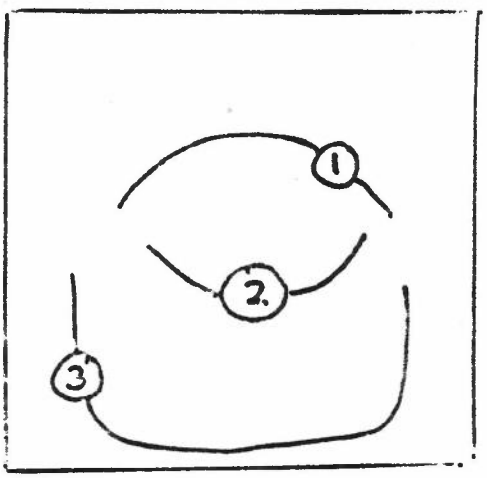
POLYLINE INDEX	LINE TYPE	LINEWIDTH SCALE FACTOR	COLOUR INDEX
1			
2	Solid	2.0	4
3			
4			
5			

POLYLINE INDEX EXAMPLE

```
SET POLYLINE INDEX (1)
POLYLINE (N, XA1, YA1)

SET POLYLINE INDEX (2)
POLYLINE (M, XA2, YA2)

SET POLYLINE INDEX (3)
POLYLINE (P, XA3, YA3)
```

Picture in NDC space has a single attribute per polyline – the polyline index.

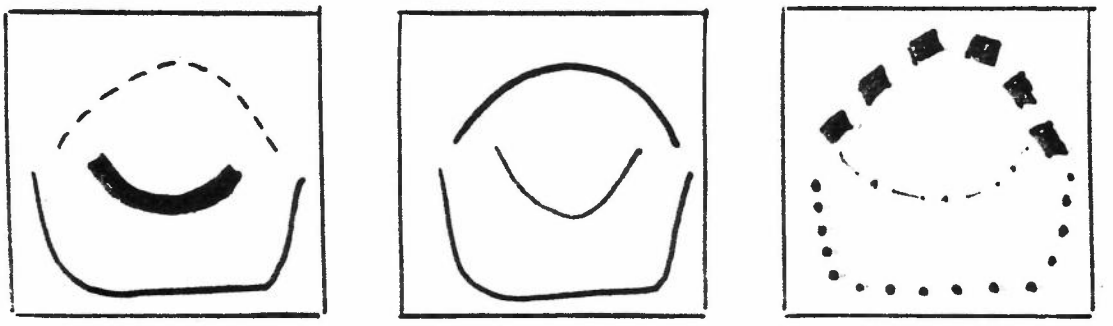
NDC Space



POLYLINE BUNDLE TABLE

Polyline Bundle Table for each workstation defines how NDC picture will appear on display surface

WORKSTATION 1 | WORKSTATION 2 | WORKSTATION 3

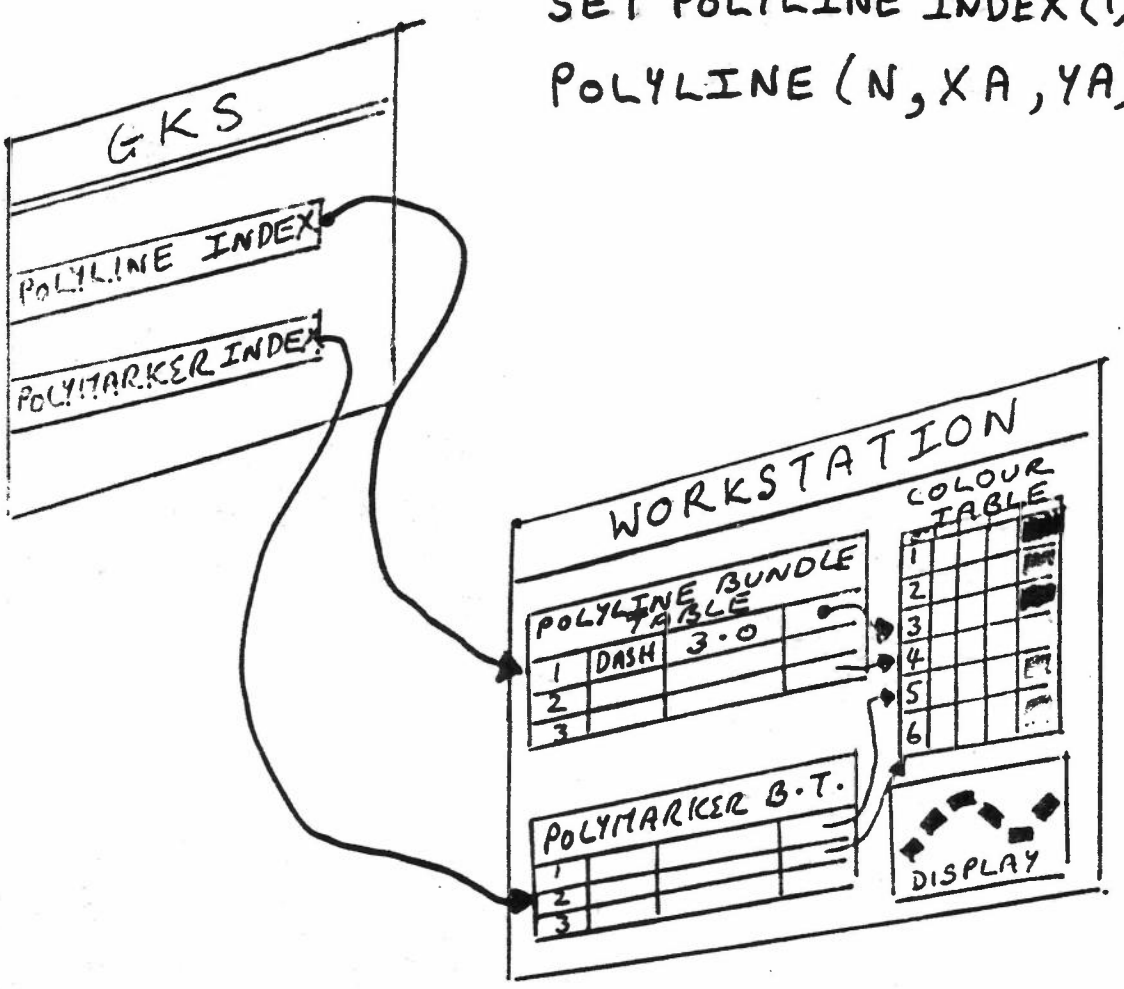


Workstation defines bundle table entries most appropriate for device in order to achieve differentiability

BUNDLED CONTROL

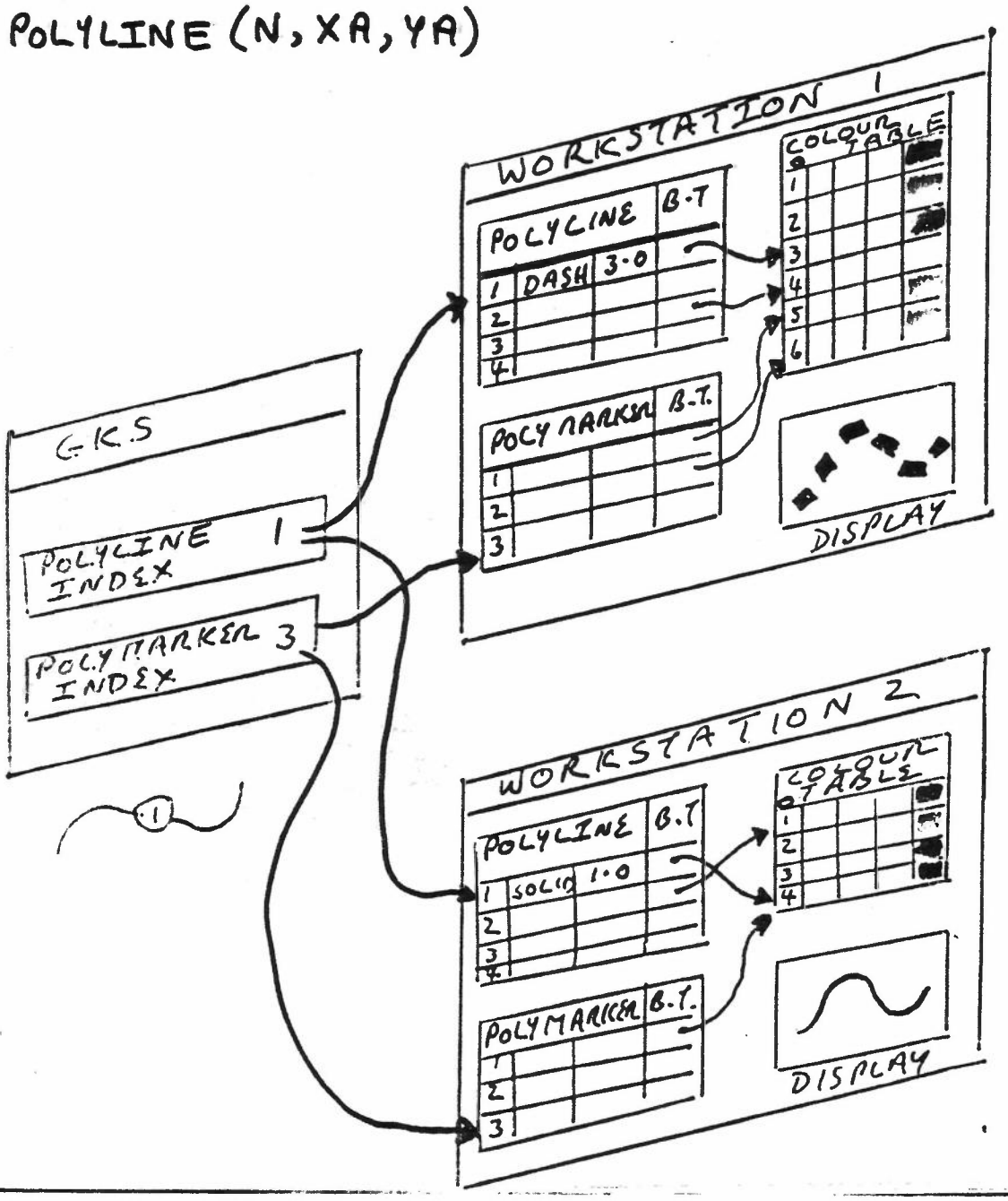
SET POLYLINE INDEX (1)

POLYLINE (N, XA, YA)



BUNDLED CONTROL

SET POLYLINE INDEX(1)
POLYLINE (N, XA, YA)



COLOUR TABLE

ENTRY	RED INTENSITY	GREEN INTENSITY	BLUE INTENSITY
0	1·0	0·0	0·0
1	0·3	0·5	1·0
2	0·0	0·0	1·0
3	0·6	0·3	0·8
4	0·0	1·0	0·0
5			

Colour Table entries define intensities in range [0, 1]

POLYMARKER BUNDLE TABLE

INDEX	MARKER TYPE	MARKERSIZE SCALE FACTOR	COLOUR INDEX
1			
2	3	2·0	2
3			
4			
5			

TEXT BUNDLE TABLE

INDEX	TEXT FONT AND PRECISION	CHARACTER EXPANSION FACTOR	CHARACTER SPACING	COLOUR INDEX
1				
2	FONT 3 'STROKE'	2·0	0·0	3
3				
4				

FILL AREA BUNDLE TABLE

INDEX	STYLE	STYLE INDEX	COLOUR INDEX
1	HOLLOW		1
2	SOLID		2
3	PATTERN	2	
4	PATTERN	1	
5	HATCH	1	3

STYLE INDEX

ENTRY	PATTERN
1	1 0 0 1
2	0 1 3 1 3 1 3 1 0

COLOUR TABLE

0
1
2
3
4
5

SETTING BUNDLE TABLE ENTRIES

SET POLYLINE REPRESENTATION

```
CALL GSPLR(WKID,INDX,LTYP,LWD,CINDX)
```

SET POLYMARKER REPRESENTATION

```
CALL GSPMR(WKID,INDX,MTYP,MSZ,CINDX)
```

SET TEXT REPRESENTATION

```
CALL GSTXR(WKID,INDX,FONT,PREC,EX,SP,CINDX)
```

SET FILL AREA REPRESENTATION

```
CALL GSFAR(WKID,INDX,INTSTY,STINDX,CINDX)
```

SET PATTERN REPRESENTATION

```
CALL GSPAR(WKID,INDX,DIMX,DIMY,
     ISC,ISR,DX,DY,CIXA)
```

COLOUR SETTING

SET COLOUR REPRESENTATION

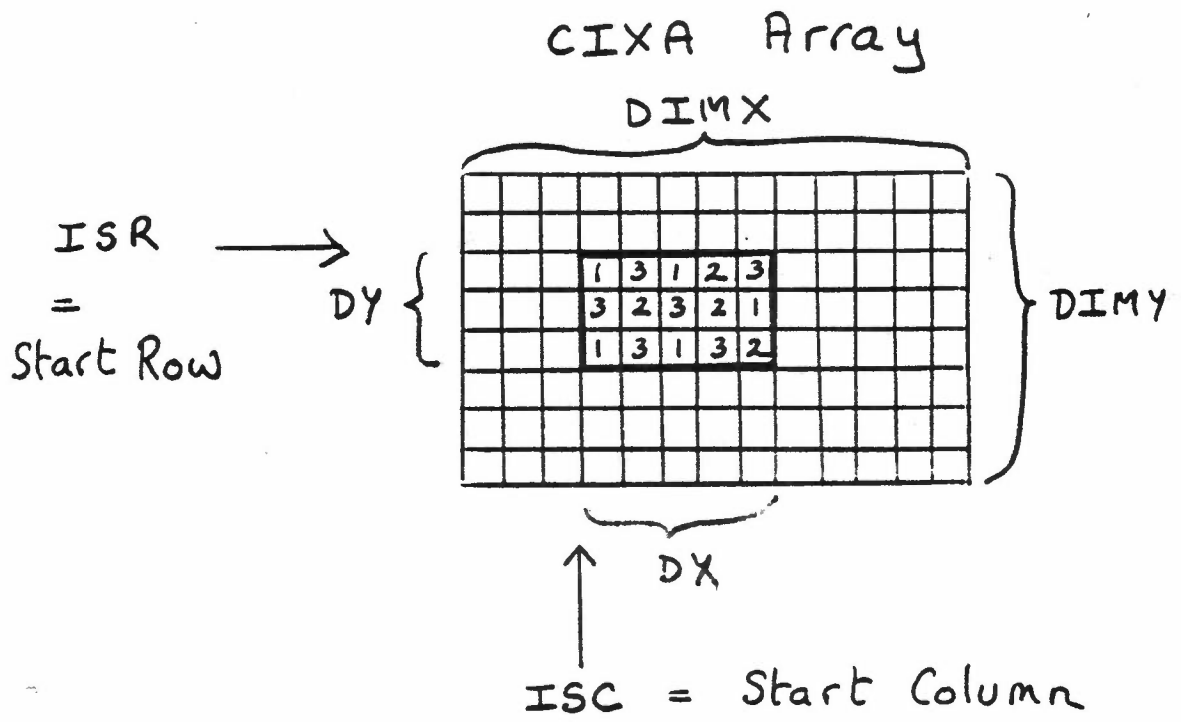
CALL GSCR (WKID, CINDX, CR, CG, CB)

Defines the CINDX row of the Colour Table on workstation WKID

CINDX = 0 defines background colour. Can be ignored by workstation if background colour cannot be changed

SET PATTERN REPRESENTATION

CALL GSPAR(WKID, INDX, DIMX, DIMY,
ISC, ISR, DX, DY, CIXA)



INDEX SETTING

SET POLYLINE INDEX
CALL GSPLI (INDX)

SET POLYMARKER INDEX
CALL GSPMI (INDX)

SET TEXT INDEX
CALL GSTXI (INDX)

SET FILL AREA INDEX
CALL GSFAI (INDX)

INDIVIDUAL CONTROL

- Each non-geometric aspect is controlled by a single attribute

- Control of appearance now defined globally rather than workstation dependent

BUNDLED

```
SET POLYLINE INDEX (1)
POLYLINE (N, XA, YA)
```

INDIVIDUAL

```
SET LINETYPE (DASHED)
SET LINEWIDTH SCALE FACTOR (2·0)
SET POLYLINE COLOUR INDEX (3)
POLYLINE (N, XA, YA)
```

POLYLINE ATTRIBUTE SETTING
INDIVIDUAL CONTROL

SET LINETYPE

CALL GSLN(LTYPE)

SET LINEWIDTH SCALE FACTOR

CALL GSLWSC(LWIDTH)

SET POLYLINE COLOUR INDEX

CALL GSPLCI(COLI)

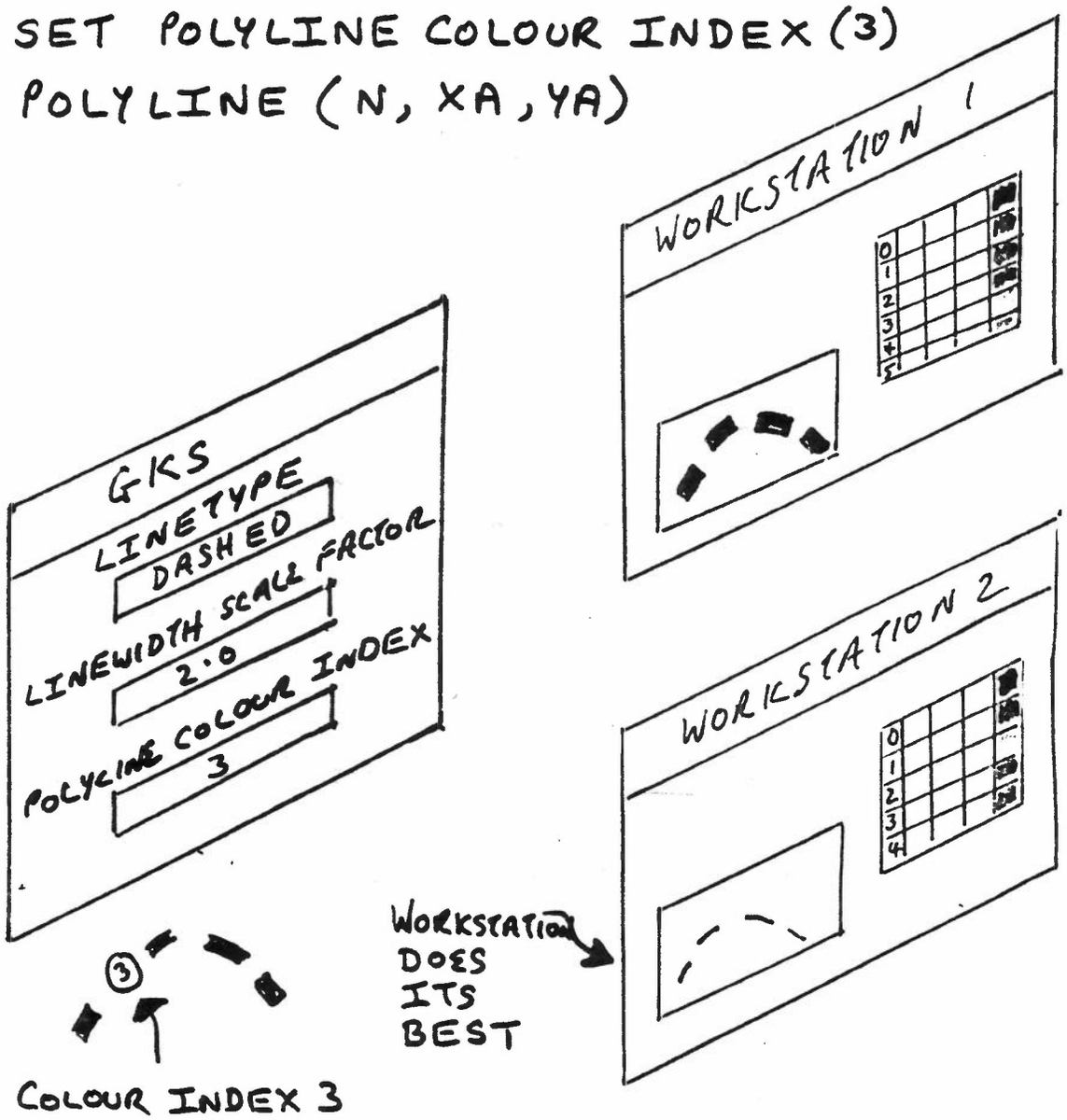
INDIVIDUAL CONTROL

SET LINETYPE (DASHED)

SET LINEWIDTH SCALE FACTOR (2·0)

SET POLYLINE COLOUR INDEX (3)

POLYLINE (N, XA, YA)



INDIVIDUAL CONTROL

- Colours will be same on different workstations only if the colour tables of workstations the same
- If application program specifies dashed thick lines, all workstations will attempt to produce them
- Attributes are modal and remain in force until reset

EXAMPLE

```
SET LINEWIDTH SCALE FACTOR (2.0)
SET LINETYPE (DASHED)
POLYLINE (NA, XA, YA)
POLYLINE (NB, XB, YB)
SET LINETYPE (SOLID)
POLYLINE (NC, XC, YC)
SET LINEWIDTH SCALE FACTOR (1.0)
POLYLINE (ND, XD, YD)
```

LINE	TYPE	WIDTH
A	DASHED	2.0
B	DASHED	2.0
C	SOLID	2.0
D	SOLID	1.0

POLYMARKER ATTRIBUTE SETTING

INDIVIDUAL CONTROL

SET MARKER TYPE

```
CALL GSMK (MTYPE)
```

SET MARKER SIZE SCALE FACTOR

```
CALL GSMKSC (MSZSF)
```

SET POLYMARKER COLOUR INDEX

```
CALL GSPMCI (COLI)
```

TEXT ATTRIBUTE SETTING

SET TEXT FONT AND PRECISION

```
CALL GSTXFP (FONT, PREC)
```

SET CHARACTER EXPANSION FACTOR

```
CALL GSCHXP (CHXP)
```

SET CHARACTER SPACING

```
CALL GSCHSP (CHSP)
```

SET TEXT COLOUR INDEX

```
CALL GSTXCI (COLI)
```

FILL AREA ATTRIBUTE SETTING INDIVIDUAL CONTROL

SET FILL AREA INTERIOR STYLE

```
CALL GSFAIS (INTS)
```

SET FILL AREA STYLE INDEX

```
CALL GSFASI (STYLI)
```

SET FILL AREA COLOUR INDEX

```
CALL GSFACI (COLI)
```

SELECTING MODE

- BUNDLED attributes is the normal default setting

- INDIVIDUAL attributes selected by :

```
SET ASPECT SOURCE FLAGS
CALL GSASF (LASF)
```

LASF is an array which allows attributes/aspects to be set individually to BUNDLED or INDIVIDUAL

LIST OF ASPECT SOURCE FLAGS

SET ASPECT SOURCE FLAGS
CALL GSASF (LASF)

Values are BUNDLED or INDIVIDUAL

1	LINETYPE
2	LINEWIDTH SCALE FACTOR
3	POLYLINE COLOUR INDEX
4	MARKER TYPE
5	MARKER SIZE SCALE FACTOR
6	POLYMARKER COLOUR INDEX
7	TEXT FONT AND PRECISION
8	CHARACTER EXPANSION FACTOR
9	CHARACTER SPACING
10	TEXT COLOUR INDEX
11	FILL AREA INTERIOR STYLE
12	FILL AREA STYLE INDEX
13	FILL AREA COLOUR INDEX

RECOMMENDATION

- Unless skilled, work either in BUNDLED or INDIVIDUAL mode, DO NOT MIX

- BUNDLED gives portability across range of diverse devices where differentiability of primitive types is important

- INDIVIDUAL is closest to current practice and gives similarity of output across range of devices

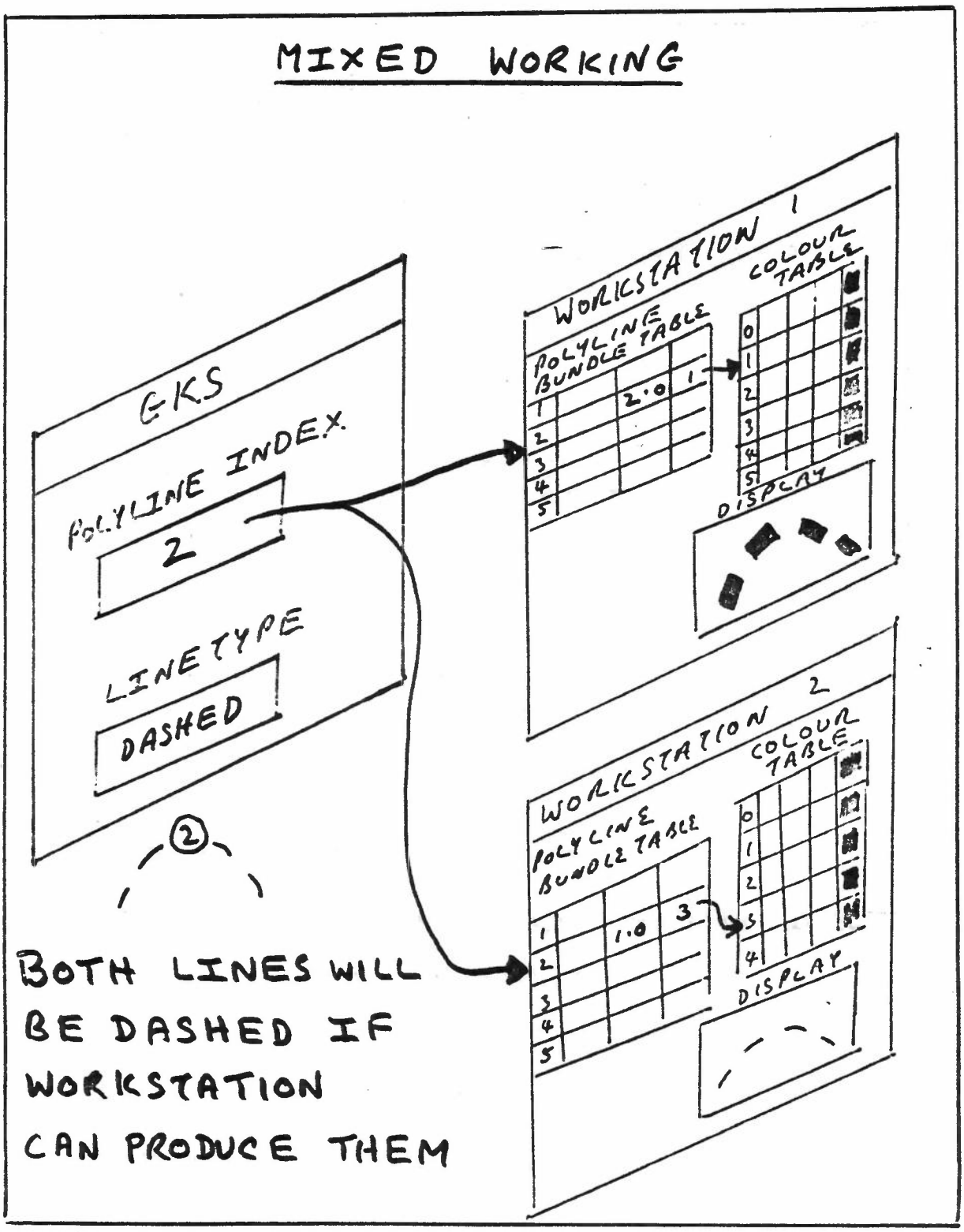
MIXED WORKING, POLYLINE EXAMPLE

- If particular aspect is defined as INDIVIDUAL, global setting is used
- If particular aspect is defined as BUNDLED, workstation uses bundle table index to locate current setting on that workstation

ASPECT SOURCE FLAGS

LINETYPE	INDIVIDUAL
LINEWIDTH SCALE FACTOR	BUNDLED
POLYLINE COLOUR INDEX	BUNDLED

```
SET POLYLINE INDEX (2)
SET LINETYPE (DASHED)
POLYLINE (N, XA, YA)
```


MIXED WORKING
GKS
POLYLINE INDEX
2
LINETYPE
DASHED
2
BOTH LINES WILL BE DASHED IF WORKSTATION CAN PRODUCE THEM
WORKSTATION 1
POLYLINE BUNDLE TABLE
COLOUR TABLE
DISPLAY
WORKSTATION 2
POLYLINE BUNDLE TABLE
COLOUR TABLE
DISPLAY

GKS INPUT

Logical input devices can be used for:

- pointing
- selecting
- sketching
- placing
- typing
- digitizing
- real time control

Facilities controllable by application:

- prompting
- echoing
- acknowledgment
- operator messages

GKS LOGICAL INPUT DEVICES

GKS defines six classes :

LOGICAL INPUT DEVICE	PROVIDES
LOCATOR	• position in world coordinates and normalization transformation number
VALUATOR	• a real number
CHOICE	• a non-negative integer representing a selection from a set of choices
PICK	• a segment name and pick identifier
STRING	• a character string
STROKE	• sequence of positions in world coordinates and normalization transformation number

Device can also return status information

OPERATING MODES

The operating modes of logical input devices specify which of operator and application program has the initiative :

REQUEST input is produced by operator in direct response to the application program

SAMPLE input is acquired directly by the application program

EVENT input is generated asynchronously by the operator and is collected in a queue for the application program

PHYSICAL INPUT DEVICES

Joystick

Lightpen

Keyboard

Mouse

Tablet

Thumb wheels

Function buttons

Dials

Tracker balls

etc

Any physical device can map onto any logical input device

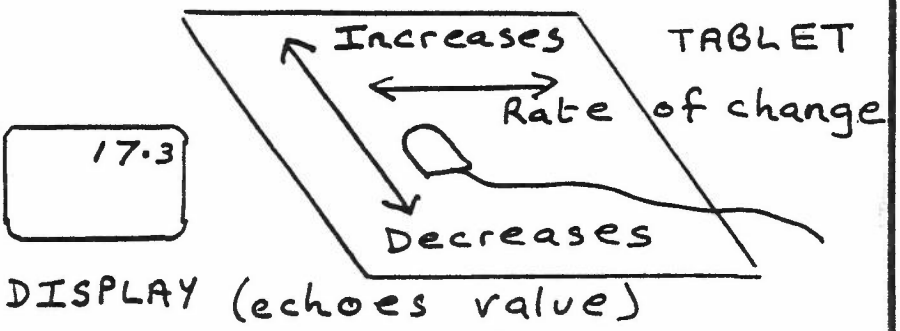
LOCATOR

Keyboard	type X, Y values
• Tablet	move pen to the position required
Lightpen	move tracking cross to position
Dial	use two to move a cursor to required position
Tracker ball	use it to define relative movements of cursor from some starting position

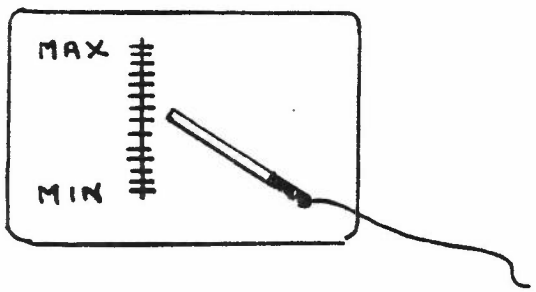
VALUATOR

Keyboard — type real number

Tablet



Lightpen

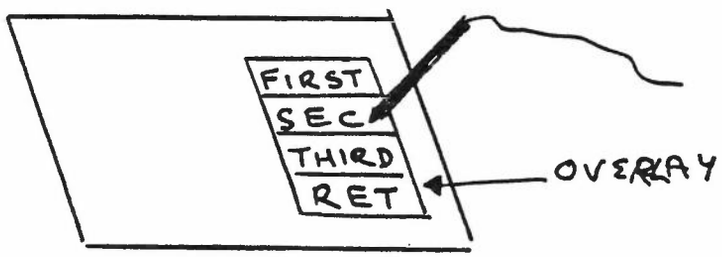


- Dial — rotate to required value

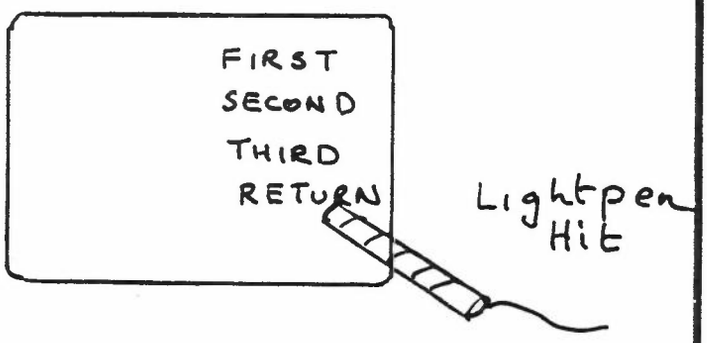
CHOICE

Keyboard — type number of required choice or name

Tablet



- Lightpen

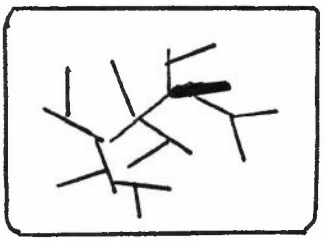


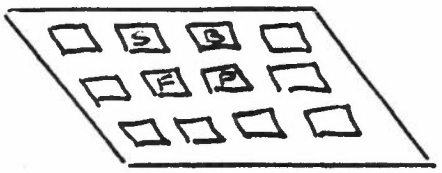
Dial — move dial with echo on display indicating current choice

PICK

Keyboard

Each primitive brightened in turn



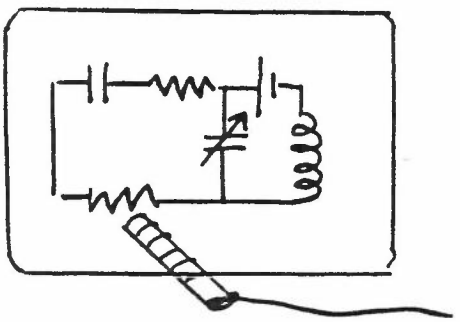


S = STOP
F = FORWARD
B = BACKWARD
P = PICK

Tablet

Tablet position input is compared with positions of primitives on display and nearest is picked

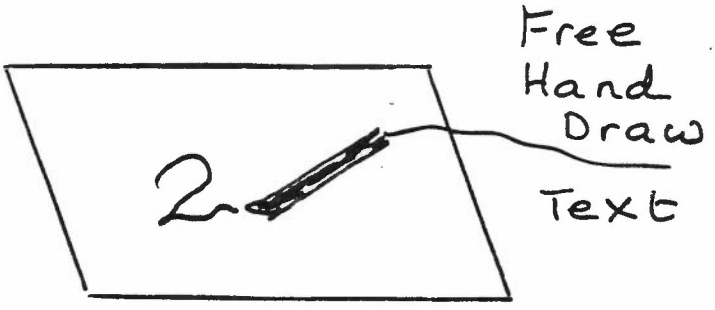
- Lightpen



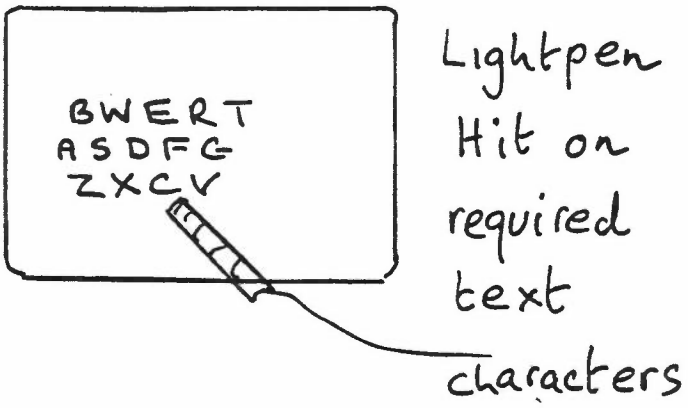
STRING

- Keyboard — type required string

Tablet



Lightpen



REQUEST MODE

- Like a FORTRAN READ
  Program suspends waiting for input. Resumes when input delivered

- Either the GKS program is executing or the logical input device is active

- Only one logical input device active at a time

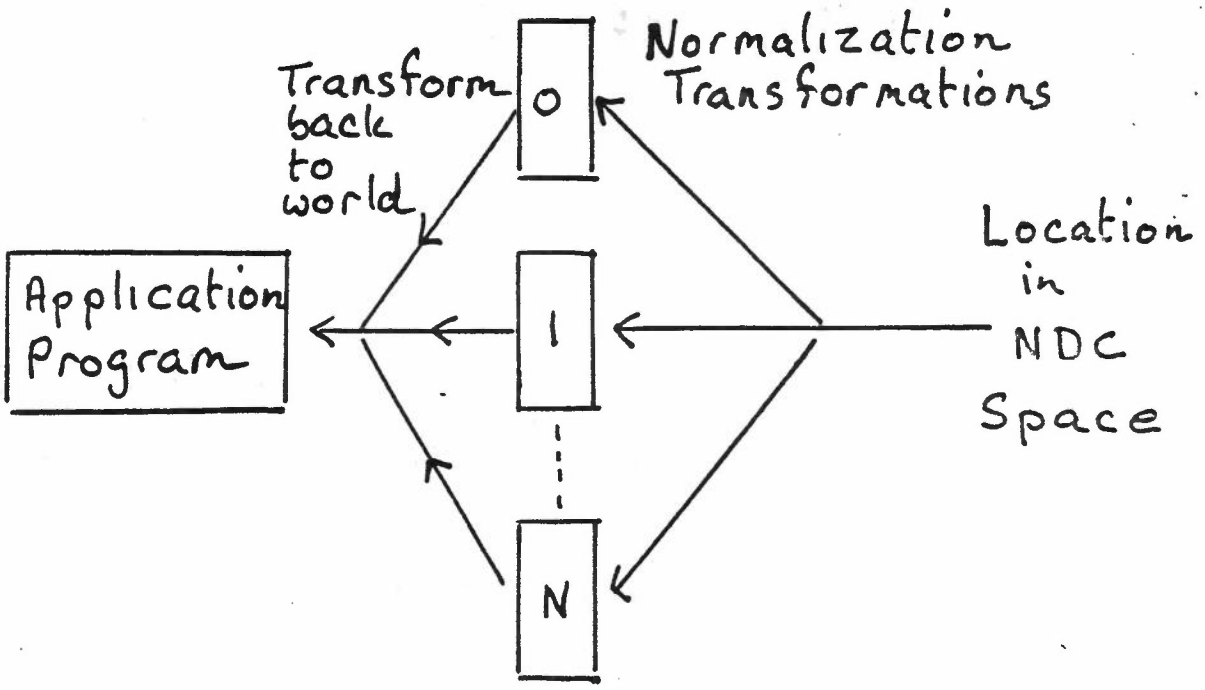
REQUEST XXX (WS, DEV, ST, ...)

WS = workstation
DEV = which device of class XXX
ST = status indicator

REQUEST LOCATOR

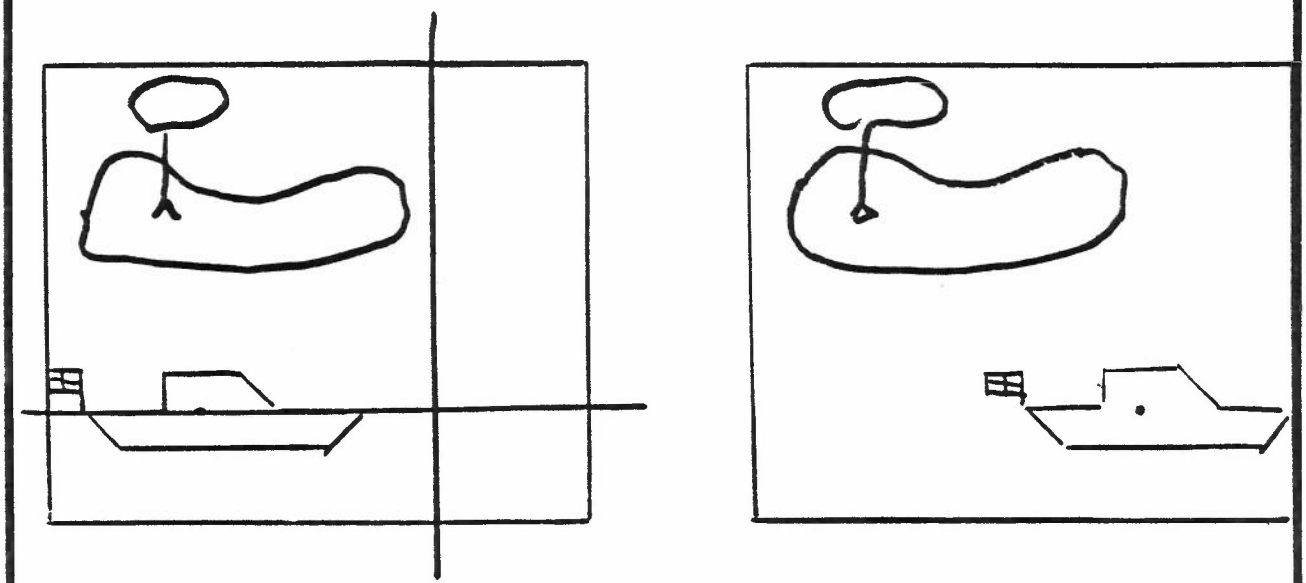
CALL GRQLC(WS, DV, ST, NORMTR, XPOS, YPOS)

Returns a position (XPOS, YPOS) in world coordinates and a normalization transformation NORMTR used to map back from NDC to world coordinates



Finds which viewport NDC position is within

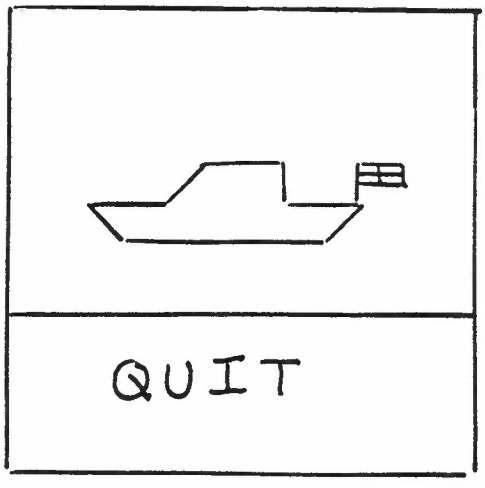
```
SET WINDOW (1, XWMIN, XWMAX, YWMIN, YWMAX)
SET VIEWPORT(1, XVMIN, XVMAX, YVMIN, YVMAX)
SELECT NORMALIZATION TRANSFORMATION (1)

100 CONTINUE
    REQUEST LOCATOR (WS, DV, ST, NT, X, Y)
    NEW FRAME
    DRAW BACKGROUND
    DRAW BOAT AT (X, Y)
    GOTO 100
```



SEVERAL VIEWPORTS

- BOAT drawn in VIEWPORT 1
- QUIT drawn in VIEWPORT 2

```
100   CONTINUE
      REQUEST LOCATOR(WS,DV,ST,NT,X,Y)
      IF (NT .EQ. 2) RETURN
      DRAW BOAT AT (X,Y)
      GOTO 100
```



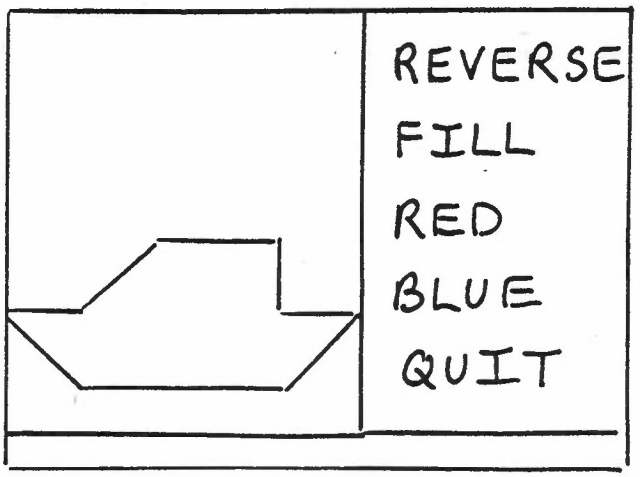
REQUEST CHOICE

CALL GRQCH(WS, DV, ST, CH)

Returns the integer CH representing the choice

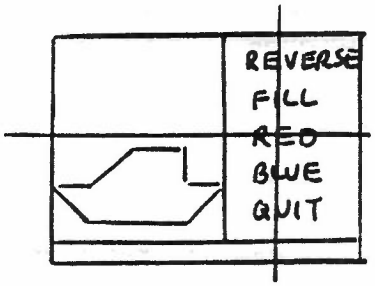
CHOICE physical device :

- Lightpen hit on a menu
- Button push
- Name typed at Keyboard
- Tablet position

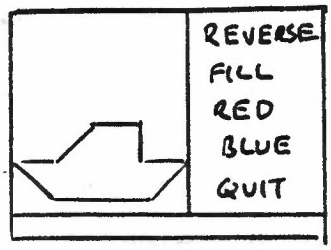



```
50  CONTINUE
    REQUEST CHOICE (WS,DV,ST,CH)
    GOTO (100,200,300,400,500),CH

100 REVERSE OBJECT
    GOTO 50

200 FILL OBJECT
    GOTO 50

300 COLOUR RED
    GOTO 50

400 COLOUR BLUE
    GOTO 50

500 STOP
```

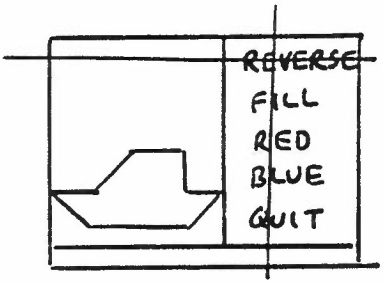
1



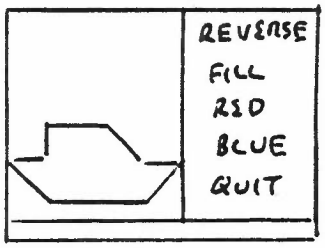
2



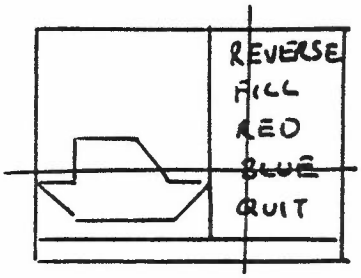
3



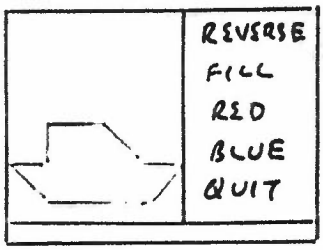
4



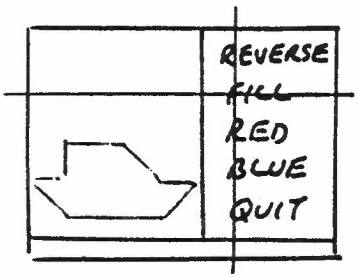
5



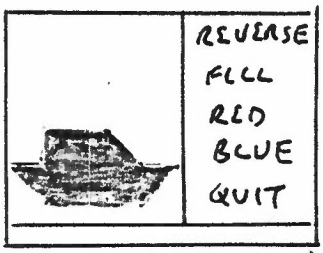
6



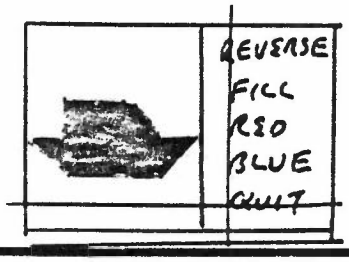
7



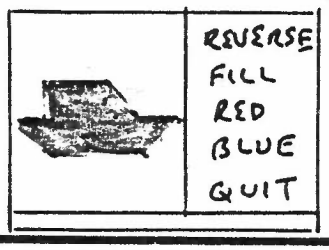
8



9



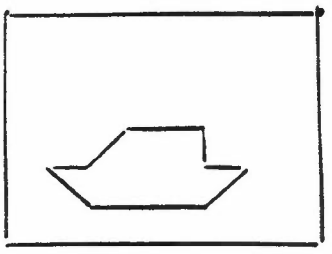
10

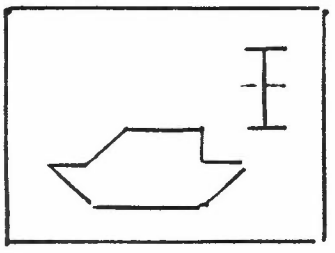


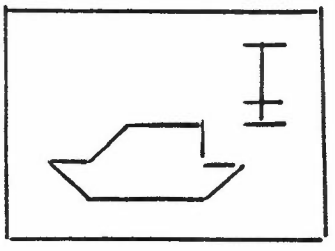
REQUEST VALUATOR

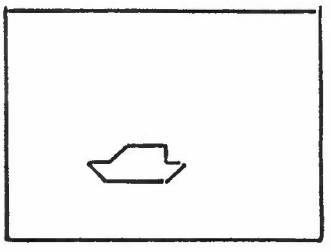
CALL GRQVL (WS, DV, ST, VAL)
Returns a real number in VAL

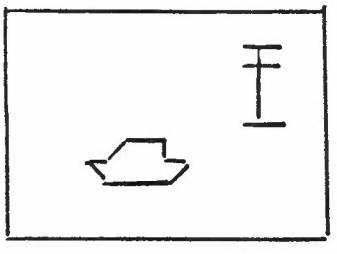
```
100  CONTINUE
     REQUEST VALUATOR (WS,DV,ST, SIZE)
     DRAW BOAT OF (SIZE)
     GOTO 100
```

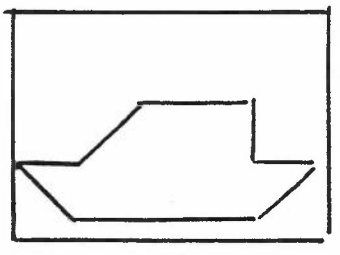
① 

② 

③ 

④ 

⑤ 

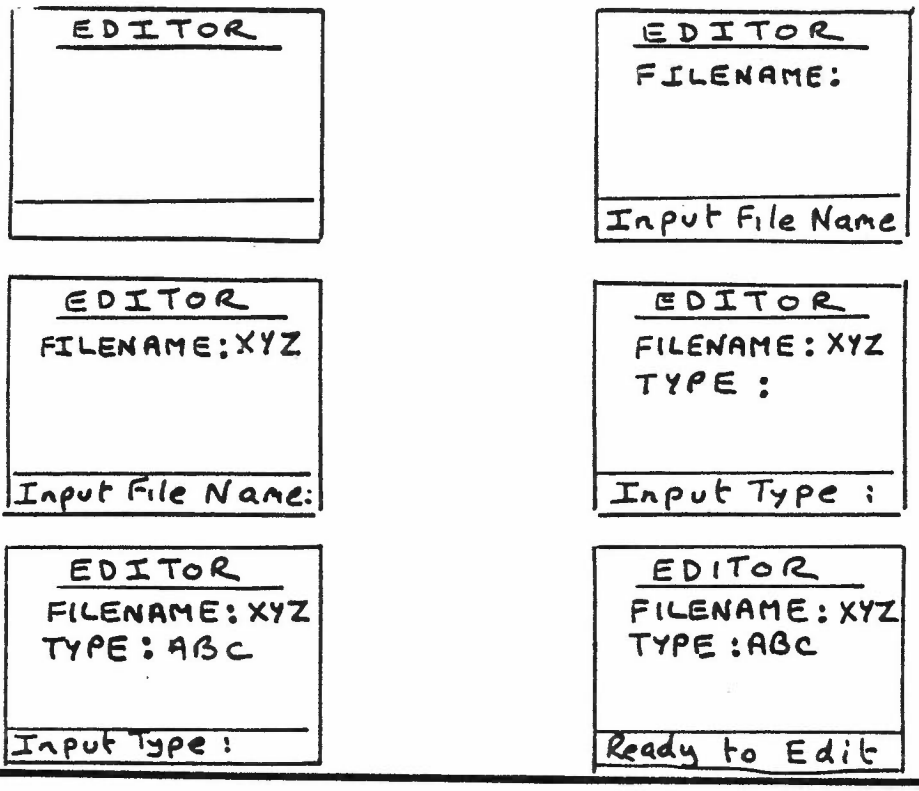
⑥ 

REQUEST STRING

CALL GRQST(WS, DV, ST, NCHARS, STR)

Reads a string of NCHARS characters into STR

REQUEST STRING(WS, DV, ST, NF, FIL)
TEXT (XF, YF, FIL)
REQUEST STRING(WS, DV, ST, NT, TYP)
TEXT (XT, YT, TYP)



REQUEST STROKE

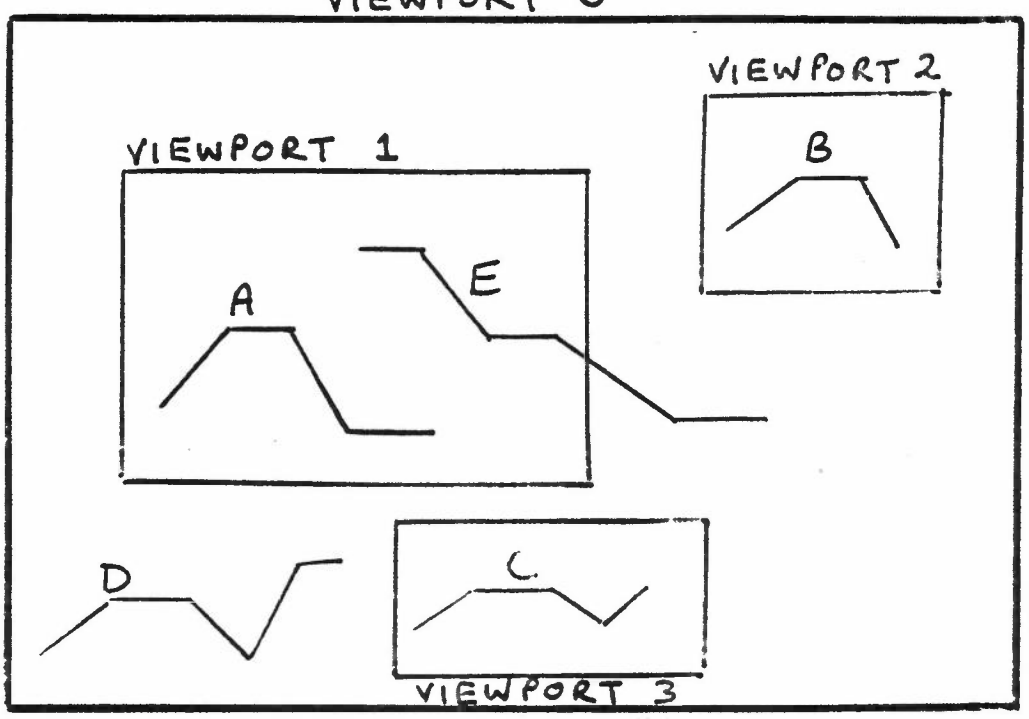
CALL GRQSK(WS, DV, PTSMAX, ST, NT, NPTS, X, Y)

Returns a sequence of NPTS positions in arrays X and Y and a normalization transformation NT used to map back from NDC to world

PTSMAX specifies the maximum number of points that can be input

All points returned are within viewport defined by NT

STROKE INPUT

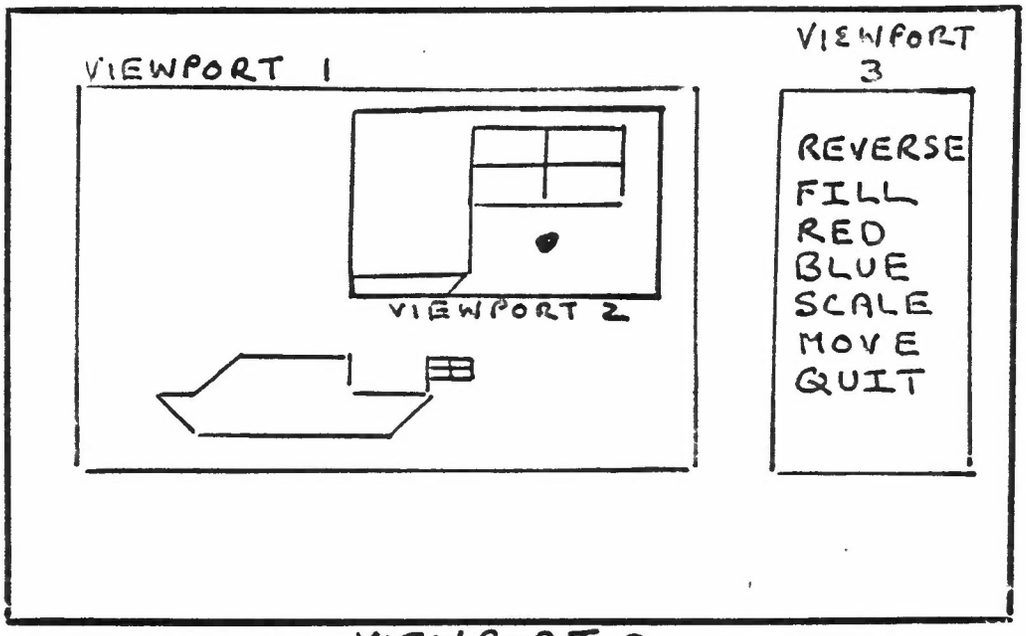


Normalization Transformation

A :	1
B :	2
C :	3
D :	0
E :	0

VIEWPORT INPUT PRIORITY

Which viewport is used to transform LOCATOR or STROKE input back to world coordinates if viewports overlap?

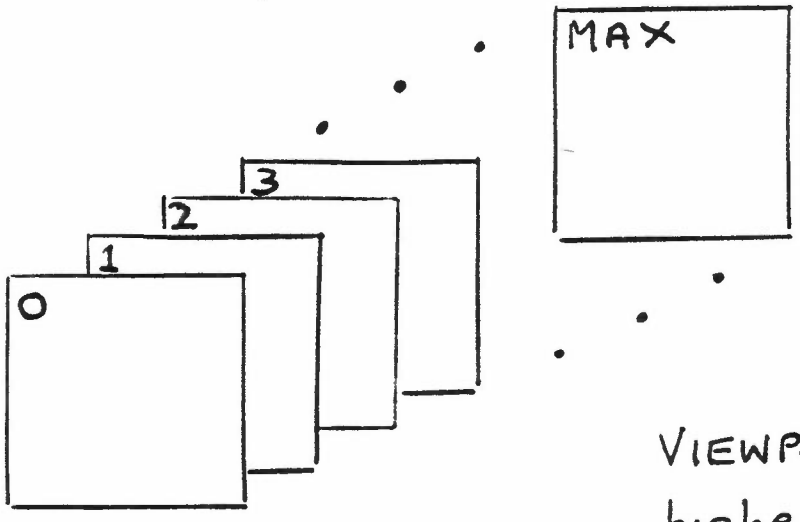


VIEWPORT INPUT PRIORITY with each transformation

Viewport with highest VIP selected if ambiguous

VIEWPORT INPUT PRIORITY

INITIAL SETTING



VIEWPORT 0 has highest VIP initially

SET VIEWPORT INPUT PRIORITY

CALL GSVPIP(TNR, RTNR, HILO)

Defines viewport input priority of TNR to be higher or lower (HILO) than the reference viewport RTNR

Priority of TNR is immediately above or below RTNR after the call

0, 1, 2, 3, 4, 5, 6

SET VIEWPORT INPUT PRIORITY (0, 6, LOWER)

1, 2, 3, 4, 5, 6, 0

SET VIEWPORT INPUT PRIORITY (5, 2, HIGHER)

1, 5, 2, 3, 4, 6, 0

SET VIEWPORT INPUT PRIORITY (2, 6, HIGHER)

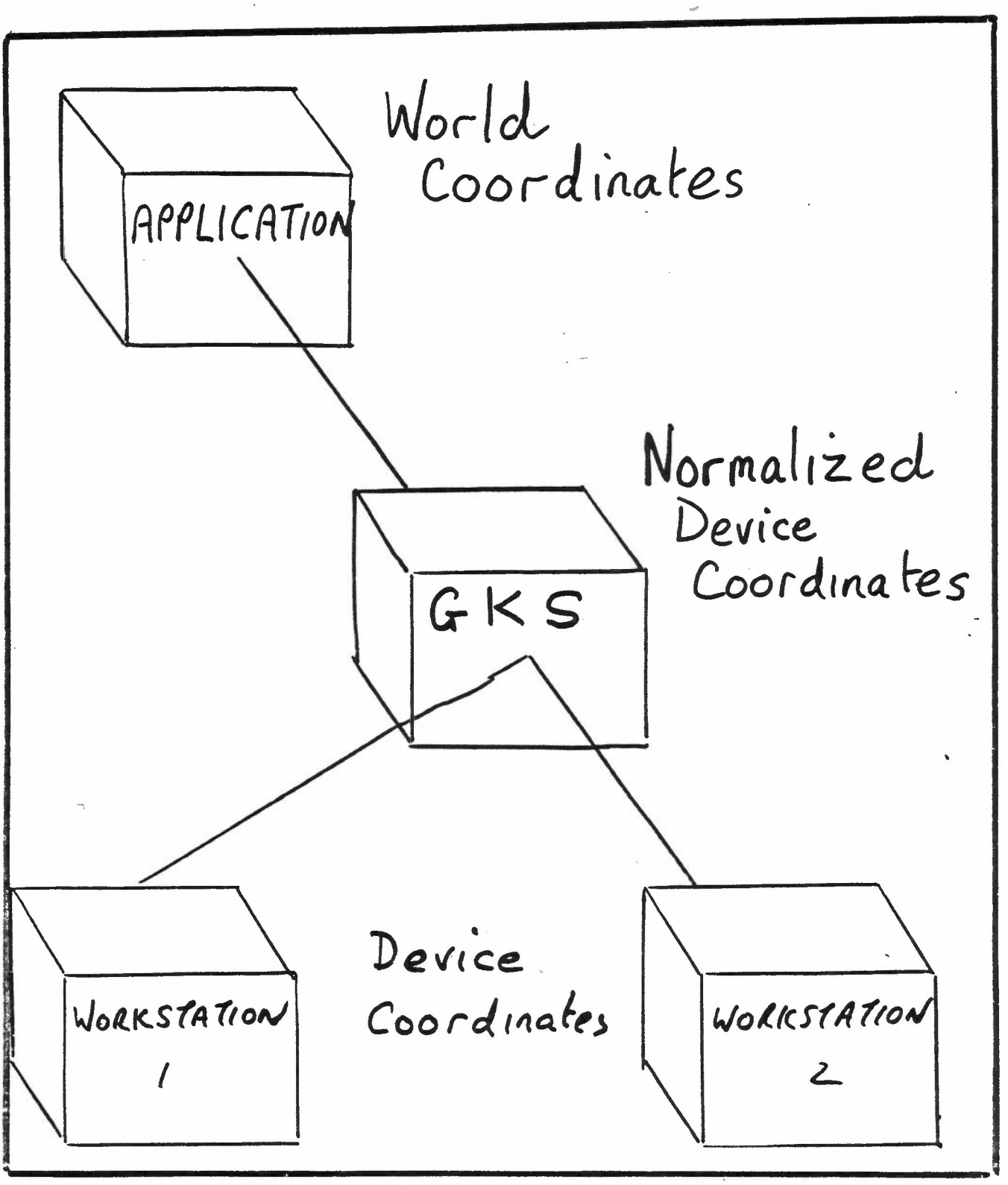
1, 5, 3, 4, 2, 6, 0

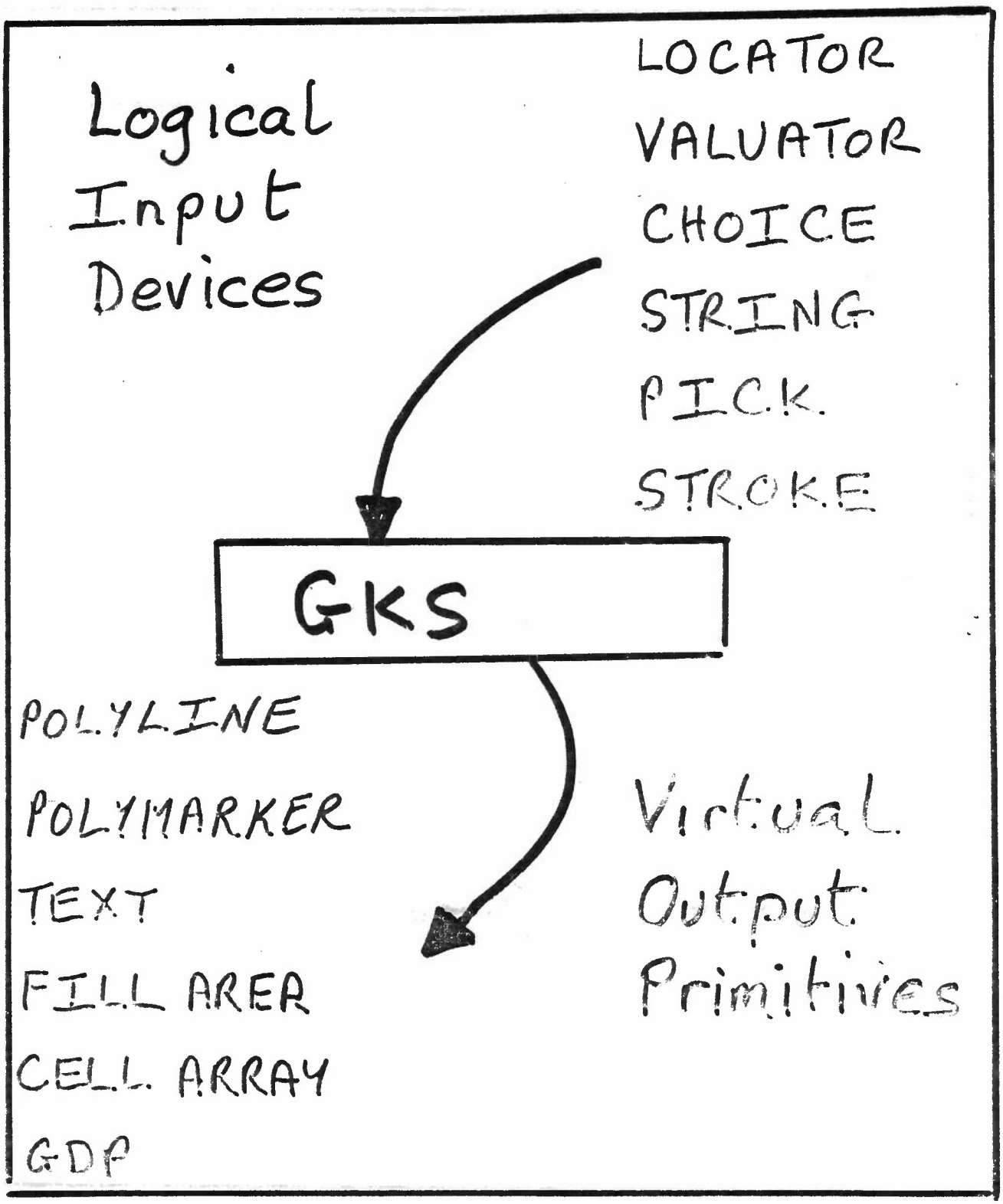
REQUEST PICK

CALL GRQPK(WS, DV, ST, SEG, PICKID)

Returns the name of a segment SEG and a more specific identification within the segment by PICKID

Detailed discussion postponed until after segments described


World
Coordinates
APPLICATION
Normalized
Device
Coordinates
GKS
Device
Coordinates
WORKSTATION
1
WORKSTATION
2


Logical
Input
Devices
LOCATOR
VALUATOR
CHOICE
STRING
PICK
STROKE
GKS
POLYLINE
POLYMARKER
TEXT
FILL AREA
CELL ARRAY
GDP
Virtual
Output
Primitives

GKS OUTPUT PRIMITIVES

POLYLINE 

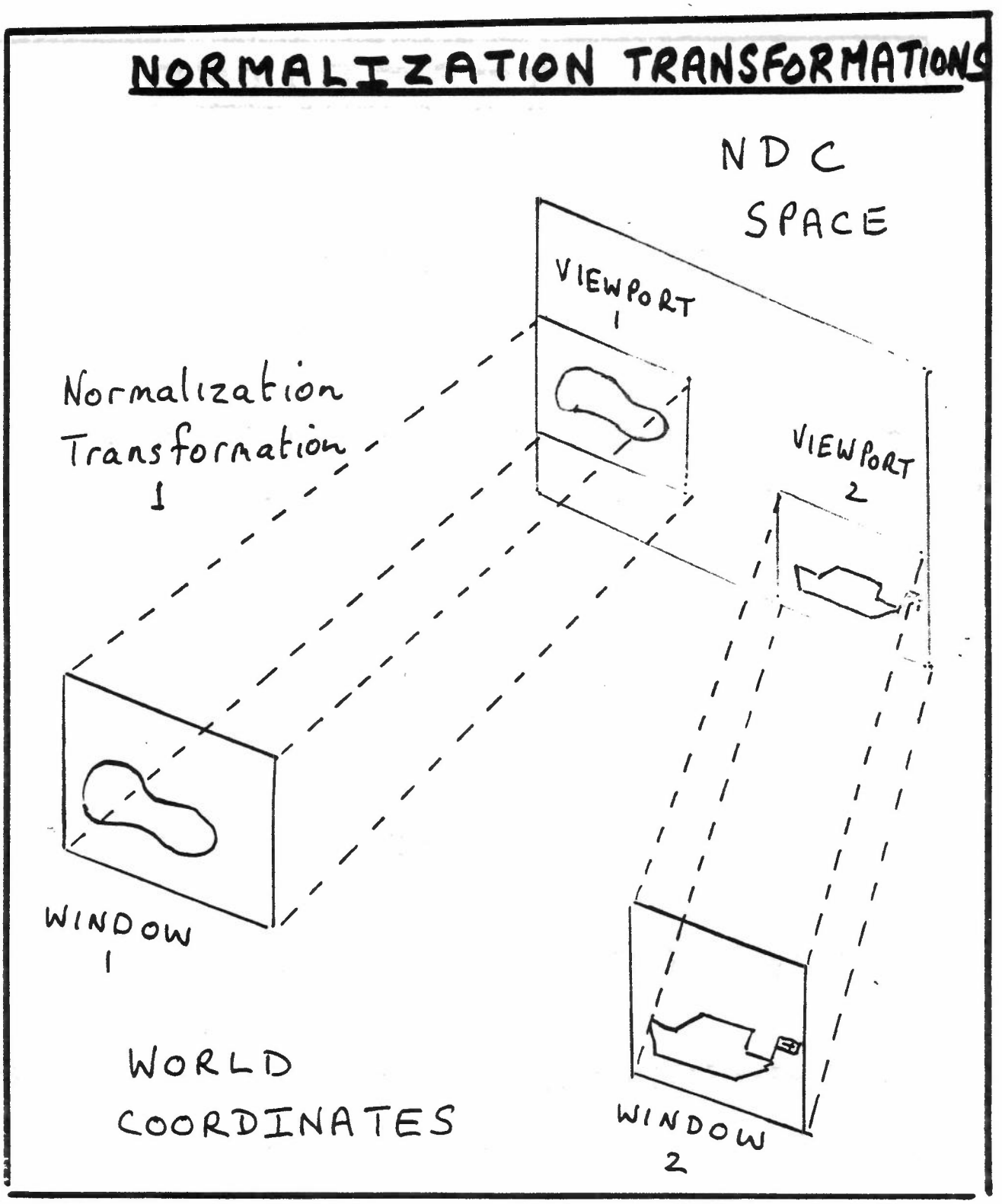
POLYMARKER 

TEXT characters

FILL AREA 

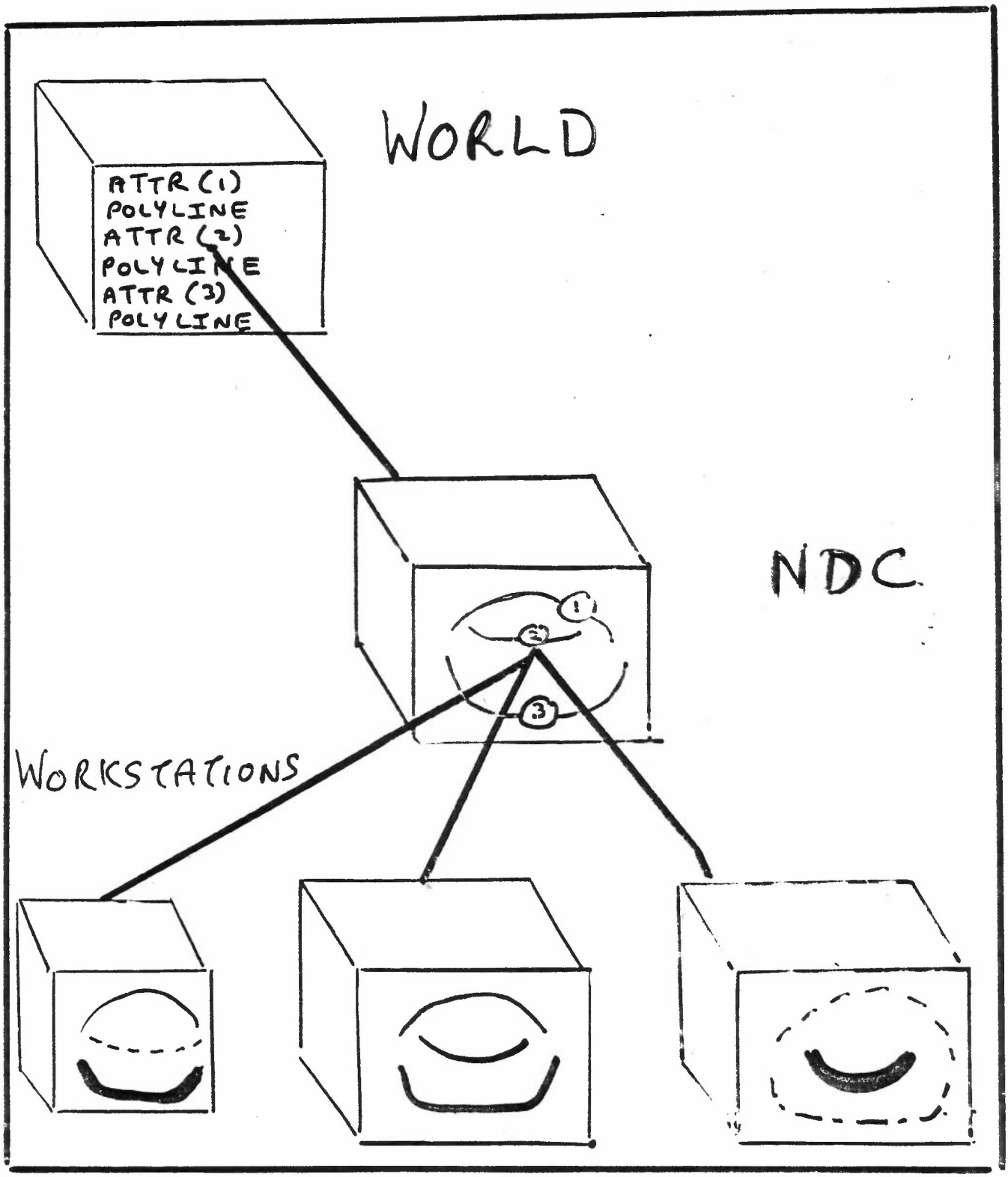
CELL ARRAY 

GDP 

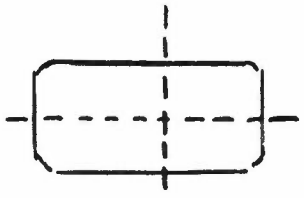
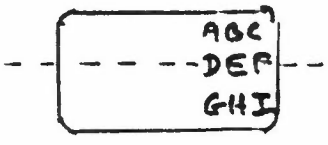
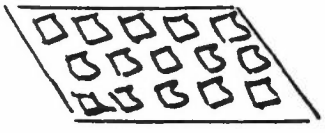
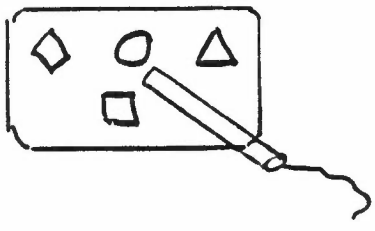
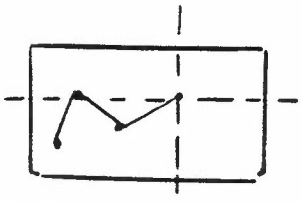

NORMALIZATION TRANSFORMATIONS
NDC
SPACE
VIEWPORT
1
VIEWPORT
2
Normalization
Transformation
1
WINDOW
1
WORLD
COORDINATES
WINDOW
2

ATTRIBUTES

- Appearance of output primitives controlled by attributes made up of one or more aspects
- Attributes may be bundled or unbundled (individual)
- Geometric attributes are workstation independent
- Non-geometric attributes defined by each workstation
- **BUNDLED** for differentiability
- **INDIVIDUAL** for faithfully representing specific aspects


WORLD
ATTR (1)
POLYLINE
ATTR (2)
POLYLINE
ATTR (3)
POLYLINE
NDC
1
2
3
WORKSTATIONS

LOGICAL INPUT DEVICES

LOCATOR	position	
VALUATOR	value	
CHOICE	choice	
STRING	string	
PICK	name	
STROKE	sequence of positions	

INPUT

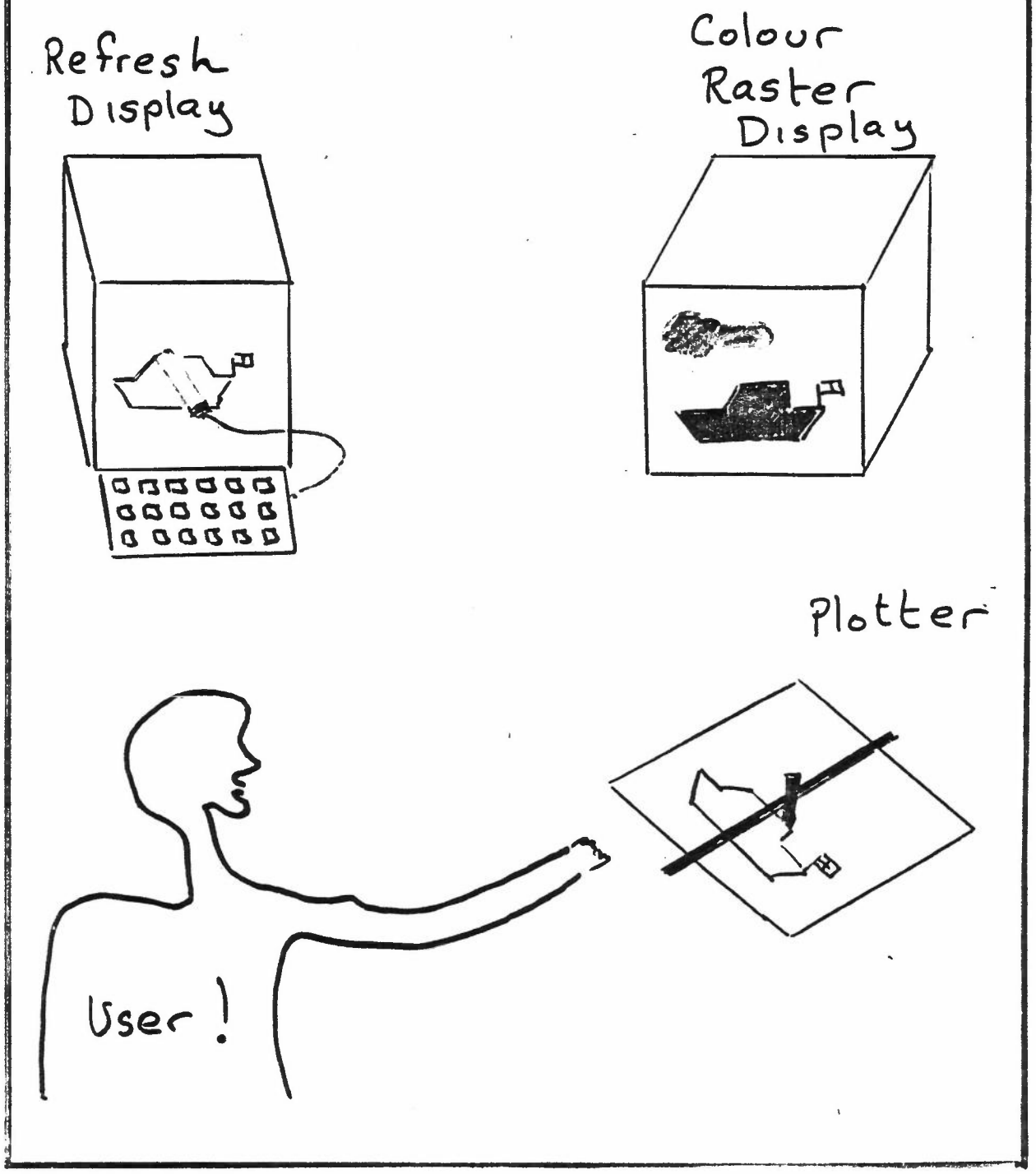
Six logical input device classes :

LOCATOR
VALUATOR
CHOICE
PICK
STRING
STROKE

Three operating modes :

REQUEST
SAMPLE
EVENT

TYPICAL USER ENVIRONMENT



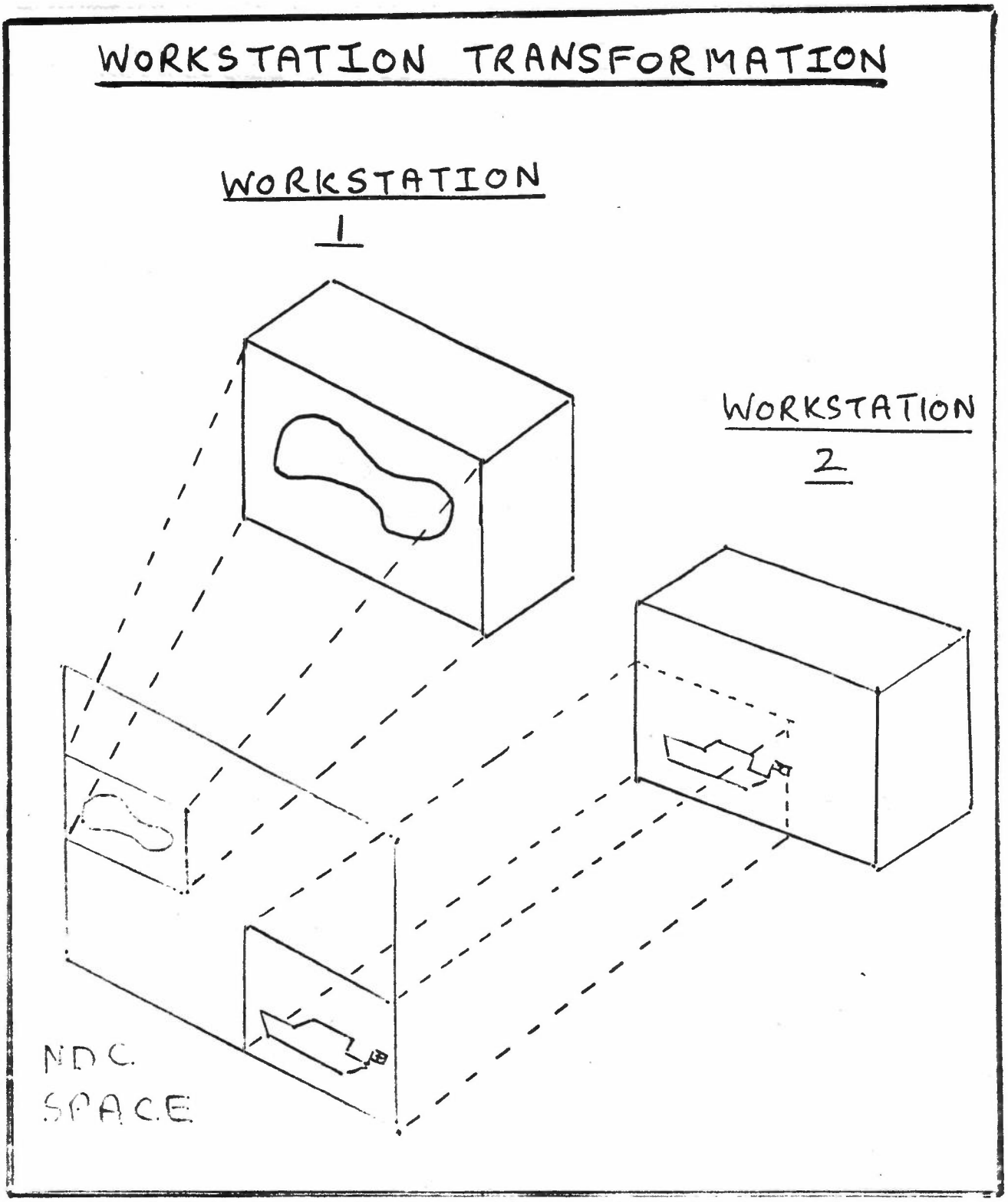
WORKSTATION CONCEPT

- The user environment consists of one or more GKS workstations
- A GKS workstation cannot have more than one display surface but it can have several input devices
- Application programs should be able to move from one user environment to another WITHOUT MAJOR CHANGES TO PROGRAM STRUCTURE

GKS WORKSTATIONS

- At most, one display surface
- Any number of input devices of every class
- A workstation transformation which defines the view of NDC space visible on the display surface
- Workstation specific primitive attributes
- Efficient Control

WORKSTATION TRANSFORMATION



WORKSTATION TRANSFORMATION

- Similar to Normalization Transformation but aspect ratio preserved
- Defines area of NDC space to be output on workstation display and position on display surface it is to appear
- Synthetic camera analogy : workstation is a camera viewing the NDC space

SET WORKSTATION WINDOW (WKID, XMN, XMX, YMN, YMX)

CALL GSWKWN (WKID, XMN, XMX, YMN, YMX)

Defines the window in NDC Space

SET WORKSTATION VIEWPORT (WSID, WS viewport)

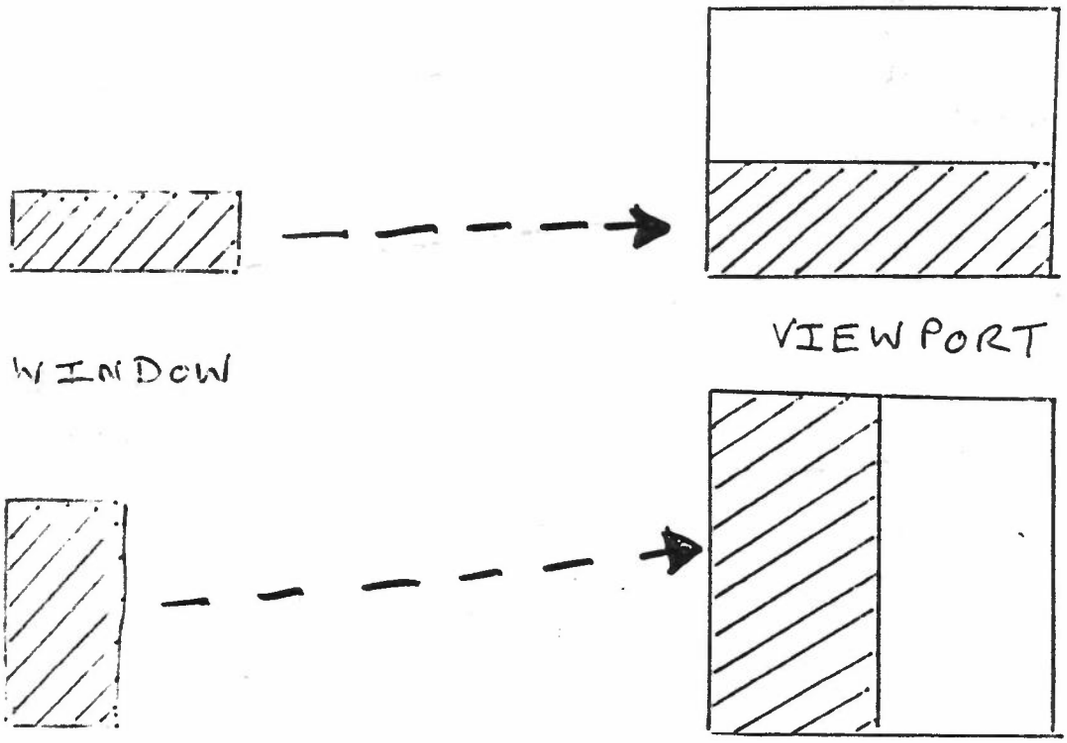
CALL GSWKVP (WKID, XVMN, XVMX, YVMN, YVMX)

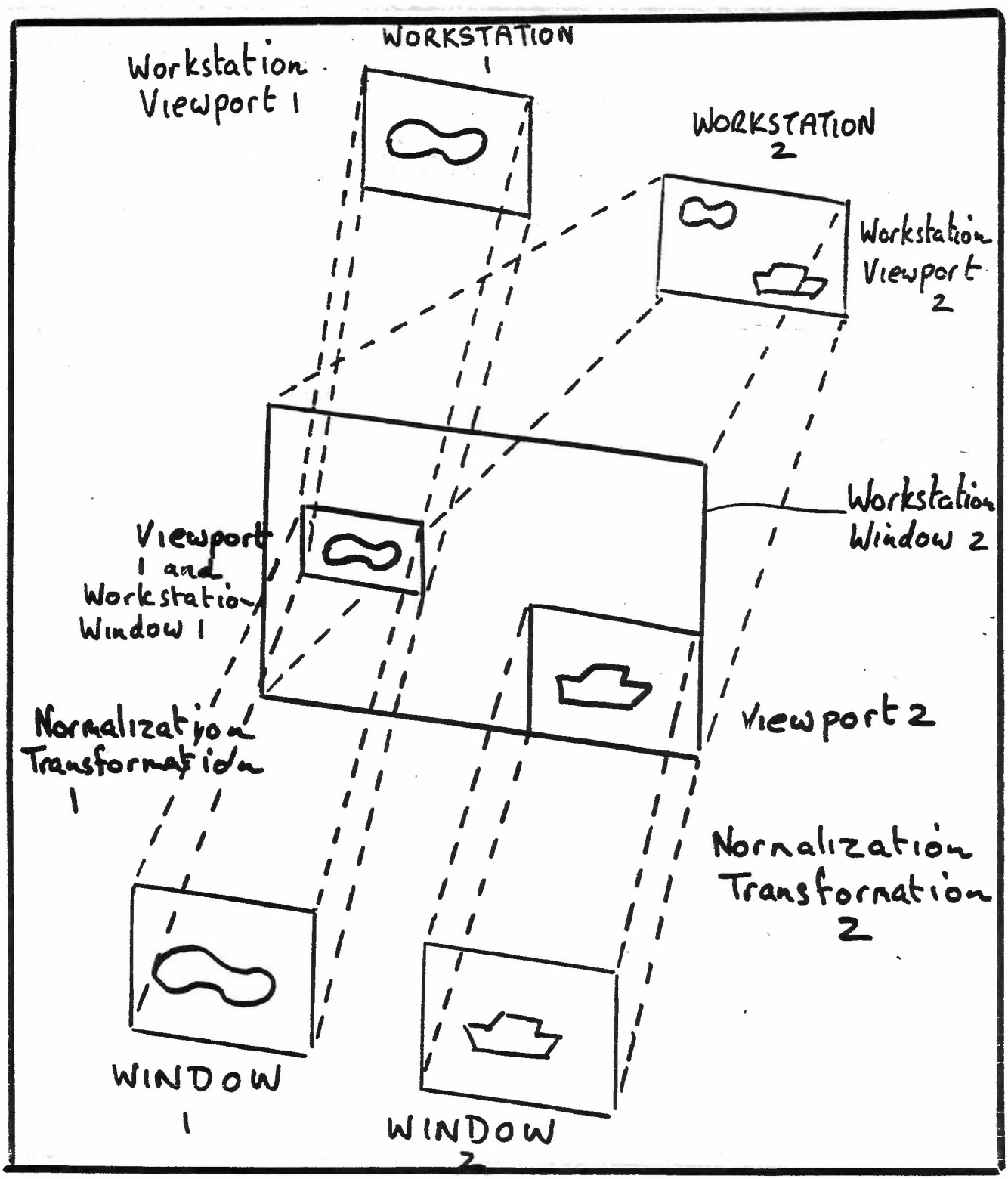
and the workstation viewport

PRESERVING ASPECT RATIO

If workstation window and viewport defined with different aspect ratios,

the complete window is displayed as large as possible with bottom left corners of window and viewport together




WORKSTATION 1
Workstation Viewport 1
WORKSTATION 2
Workstation Viewport 2
Workstation Window 2
Viewport 1 and Workstation Window 1
Viewport 2
Normalization Transformation 1
Normalization Transformation 2
WINDOW 1
WINDOW 2

WORKSTATION SELECTION

OPEN WORKSTATION

CALL GOPWK(WKID, CONID, WSTY)

Workstation with name WKID is opened on stream CONID. The workstation type is WSTY

ACTIVATE WORKSTATION

CALL GACWK(WKID)

Output primitives are sent to, and segments stored on all active workstations

DEACTIVATE WORKSTATION

CALL GDAWK(WKID)

CLOSE WORKSTATION

CALL GCLWK(WKID)

SEGMENTATION

GKS provides a segmentation facility for storing output primitives and their attributes

- Allows naming of picture parts
- Provides picture manipulation via segment attributes
- Provides picture transmission between GKS and workstations
- Allows picture redrawing
- Only one level of segmentation
- A set of primitives within a segment can be named

SEGMENT CREATION

CREATE SEGMENT (N)
CALL GCRSG(N)

Segment N is defined as the open segment. All subsequent output primitives are collected into this segment until the segment is closed.

The segment is stored on all active workstations

CLOSE SEGMENT
CALL GCLSG

The open segment is closed. Primitives may no longer be added to this segment.
The segment cannot be reopened

SEGMENTS

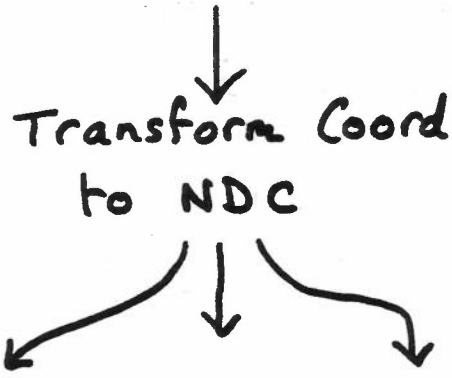
Only one segment open at a time

```
CREATE SEGMENT (1)
 OUTPUT PRIMITIVES

NOT ALLOWED →  CREATE SEGMENT (2)
 OUTPUT PRIMITIVES

CLOSE SEGMENT

CLOSE SEGMENT
```

SEGMENT CONTENTS

SET CLIPPING INDICATOR (CLIP)
SET WINDOW (N, XWMN, XWMX, YWMN, YWMX)
SET VIEWPORT (N, XVMN, XVMX, YVMN, YVMX)
SELECT NORMALIZATION TRANSFORMATION (N)
CREATE SEGMENT (S)
POLYLINE (M, XA, YA)
CLOSE SEGMENT

POLYLINE (M, XA, YA)



M	XA	YA	XVMN	XVMX	YVMN	YVMX

Stores on each active workstation polyline coordinates in unclipped NDC together with CLIPPING RECTANGLE

CLIPPING RECTANGLE = C, !, C, !
if NOCLIP

SEGMENT OPERATIONS

RENAME SEGMENT

CALL GRENSG(OLD,NEW)

Changes segment name from OLD to NEW

DELETE SEGMENT

CALL GDSG(SGNA)

The segment SGNA is deleted from all workstations

DELETE SEGMENT FROM WORKSTATION

CALL GDSGWK(WKID,SGNA)

The segment SGNA is deleted from workstation WKID

SEGMENT ATTRIBUTES

SET VISIBILITY

CALL GSVIS (SGNA, VIS)

Set segment to VISIBLE or INVISIBLE

SET HIGHLIGHTING

CALL GSHLIT (SGNA, HIL)

Set segment to HIGHLIGHTED or NORMAL

SET DETECTABILITY

CALL GSDTEC (SGNA, DET)

Set segment to DETECTABLE or UNDETECTABLE

SET SEGMENT TRANSFORMATION

CALL GSSGT (SGNA, MA)

Segment coordinates are transformed by 2×3 matrix MA

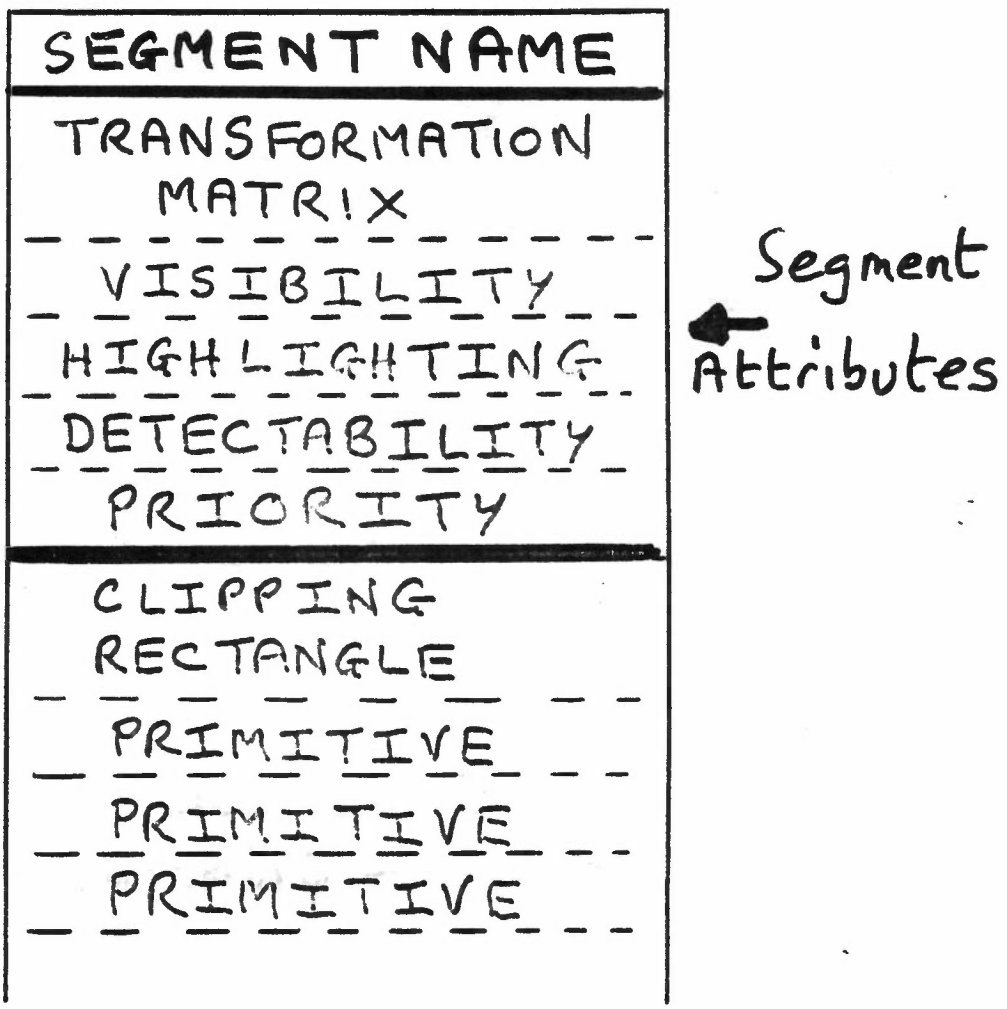
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} MA_{11} & MA_{12} & MA_{13} \\ MA_{21} & MA_{22} & MA_{23} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

EXAMPLE : RENAME SEGMENT

Boat to be drawn many times after modification

```
    I=1
    CREATE SEGMENT (OLD)
     BOAT VERSION (I)
     CLOSE SEGMENT

100  CONTINUE
     CREATE SEGMENT (NEW)
     I=I+1
     SET VISIBILITY (NEW, INVISIBLE)
     BOAT VERSION(I)
     CLOSE SEGMENT
     SET VISIBILITY (NEW, VISIBLE)
     SET VISIBILITY (OLD, INVISIBLE)
     RENAME SEGMENT (NEW
     DELETE SEGMENT (OLD)
     RENAME SEGMENT (NEW, OLD)
     GOTO 100
```

SEGMENT STRUCTURE



SEGMENT PRIORITY

SET SEGMENT PRIORITY

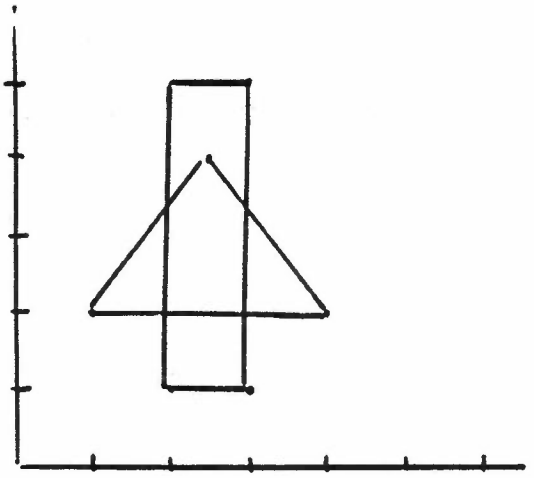
CALL GSSGP(SGNA, PRIOR)

Sets segment priority to a value in range [0, 1]

- Decides which segment obscures another
- Decides which segment PICKed by input device if segments overlap

Workstations can decline to implement – primarily aimed at raster displays

SEGMENT PRIORITY



N	XA	YA	XB	YB
1	2	1	1	2
2	2	5	2·5	4
3	3	5	4	2
4	3	1	1	2
5	2	1		

```
CREATE SEGMENT(1)
SET FILL AREA INDEX (4)      ← RED SOLID
FILL AREA (5,XA,YA)
CLOSE SEGMENT

CREATE SEGMENT(2)
SET FILL AREA INDEX(5) ← BLUE SOLID
FILL AREA (4, XB, YB)
CLOSE SEGMENT
```

SET SEGMENT PRIORITY (1, 0·5)

SET SEGMENT PRIORITY (2, 0·25)

SET SEGMENT PRIORITY (2, 0·75)

TRANSFORMATION MATRIX

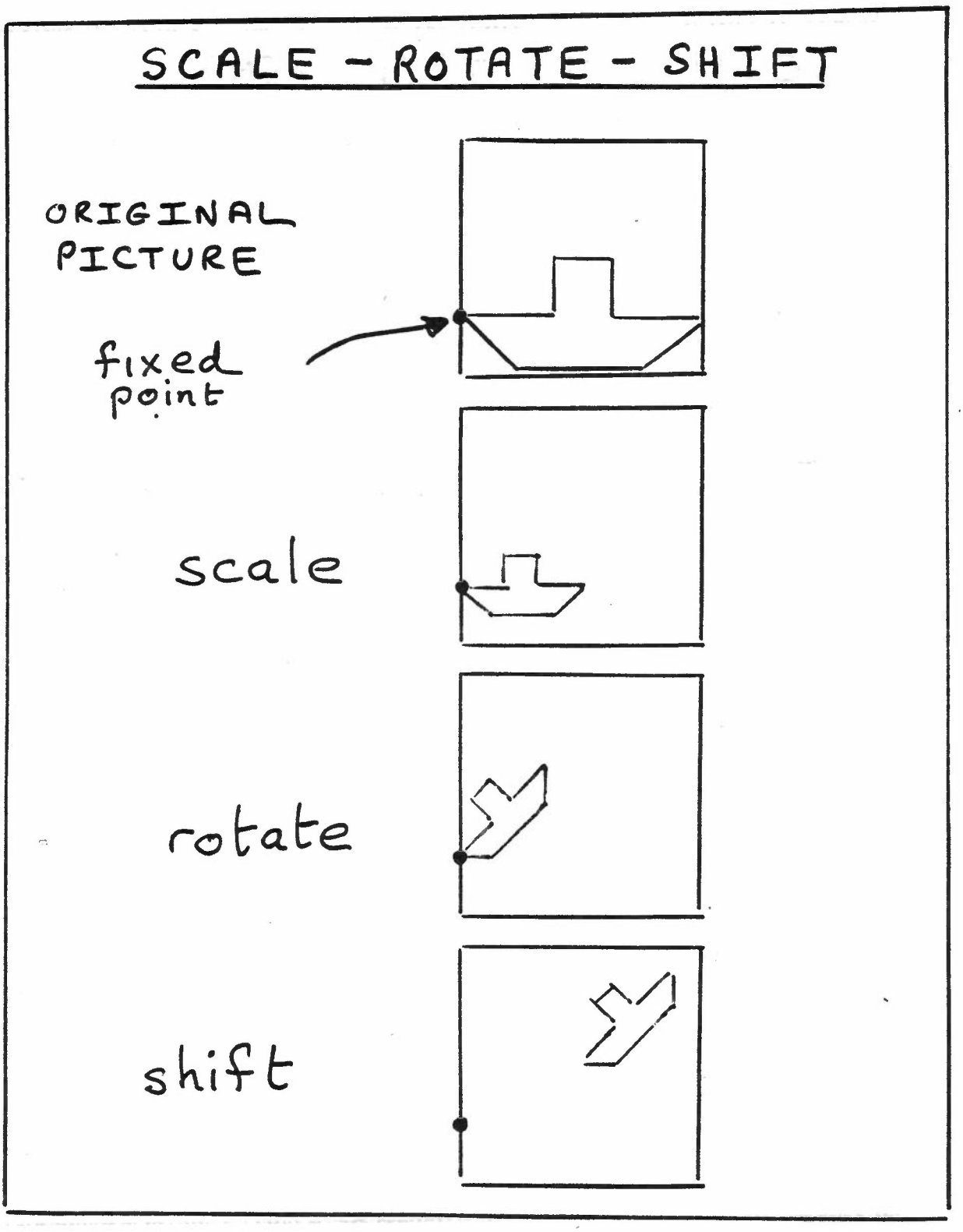
Two utility functions help in the definition of segment transformations

EVALUATE TRANSFORMATION MATRIX.

```
CALL GEVTM(XO,YO,DX,DY,PHI,FX,FY,
           SW,MOUT)
```

XO,YO	fixed point, origin for scaling and rotation
DX,DY	shift vector
PHI	rotation angle in radians (positive for anti-clockwise)
FX,FY	scale factors
SW	switch determines whether shift vector, fixed point in WC or NDC
MOUT	resulting transformation matrix

ORDER IS SCALE, ROTATE, SHIFT


SCALE - ROTATE - SHIFT
ORIGINAL PICTURE
fixed point
scale
rotate
shift

ACCUMULATE TRANSFORMATION MATRIX

```
CALL GACTM(MIN, Xo, Yo, DX, DY, PHI, FX, FY,
                                SW, MOUT)
```

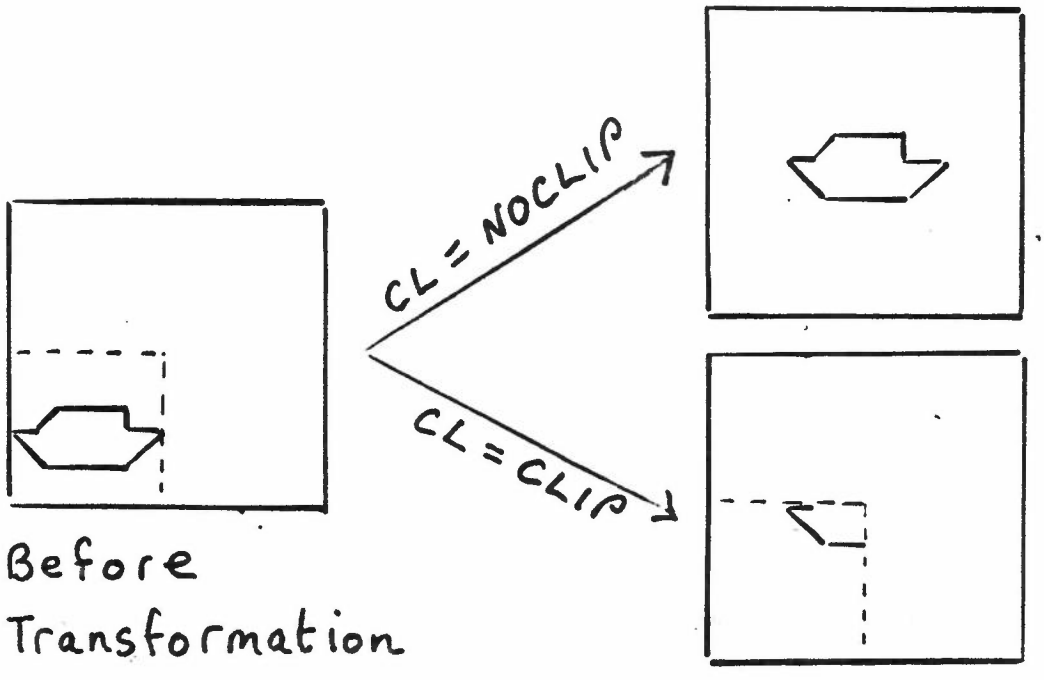
Similar to evaluation of transformation matrix

The original transformation matrix MIN is augmented by the scale, rotate and shift defined by the parameters to produce MOUT

SEGMENT TRANSFORMATION AND CLIPPING

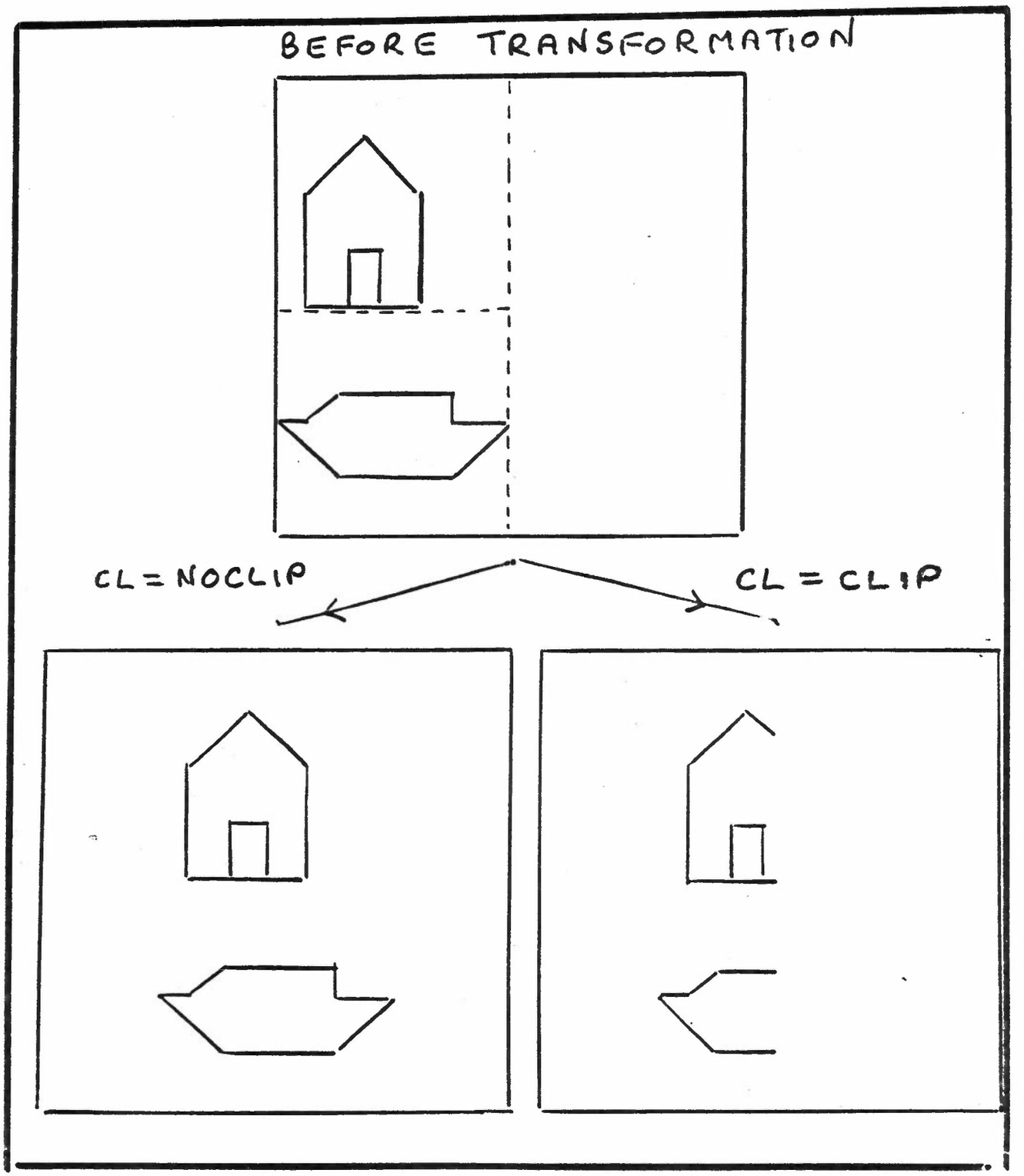
- Normalization Transformation
- Primitives stored in Segment in NDC Coordinates with clipping rectangle
- Segment Transformation
- Clipping

```
SET WINDOW (1, 0, 30, 0, 30)
SET VIEWPORT (1, 0, 0·5, 0, 0·5)
SELECT NORMALIZATION TRANSFORMATION (1)
SET CLIPPING INDICATOR (CL)
CREATE SEGMENT (11)
BOAT
CLOSE SEGMENT
EVALUATE TRANSFORMATION MATRIX (
   0, 0, 0·25, 0·25, 0, 1, 1, NOC, MAT)
SET SEGMENT TRANSFORMATION (11, MAT)
```



Before Transformation

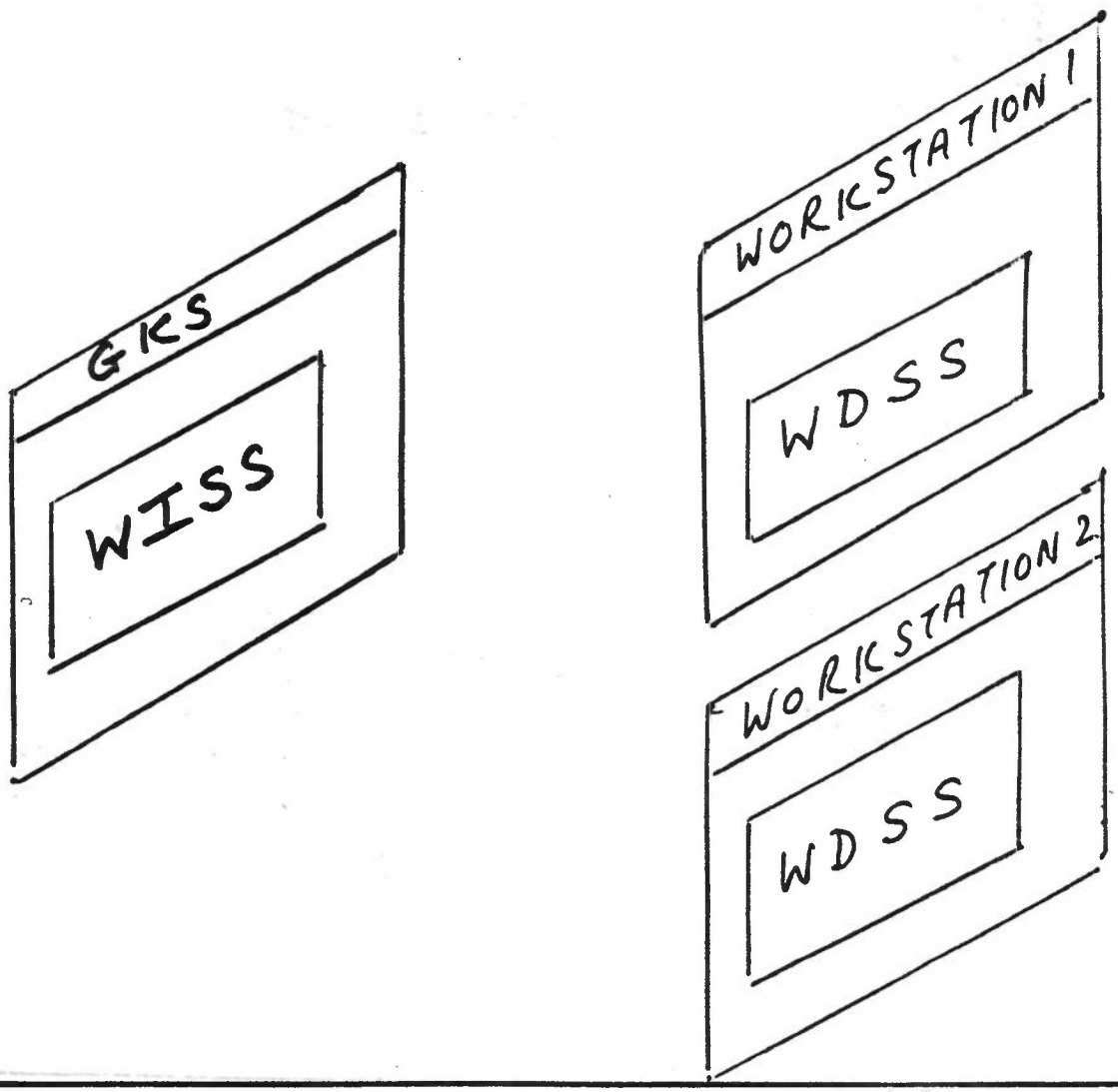
```
SET WINDOW (1, 0, 30, 0, 30)
SET VIEWPORT ( 1, 0, 0·5, 0, 0·5)
SET WINDOW (2, 0, 70, 0, 70)
SET VIEWPORT (2, 0, 0·5, 0·5, 1)

SET CLIPPING INDICATOR (CL)
CREATE SEGMENT (11)
SELECT NORMALIZATION TRANSFORMATION (1)
BOAT
SELECT NORMALIZATION TRANSFORMATION (2)
HOUSE
CLOSE SEGMENT
EVALUATE TRANSFORMATION MATRIX
   (0, 0, 0·25, 0, 0, 1, 1, NDC, MAT)
SET SEGMENT TRANSFORMATION (11, MAT)
```


BEFORE TRANSFORMATION
CL = NOCLIP
CL = CLIP

WORKSTATION INDEPENDENT SEGMENT STORAGE

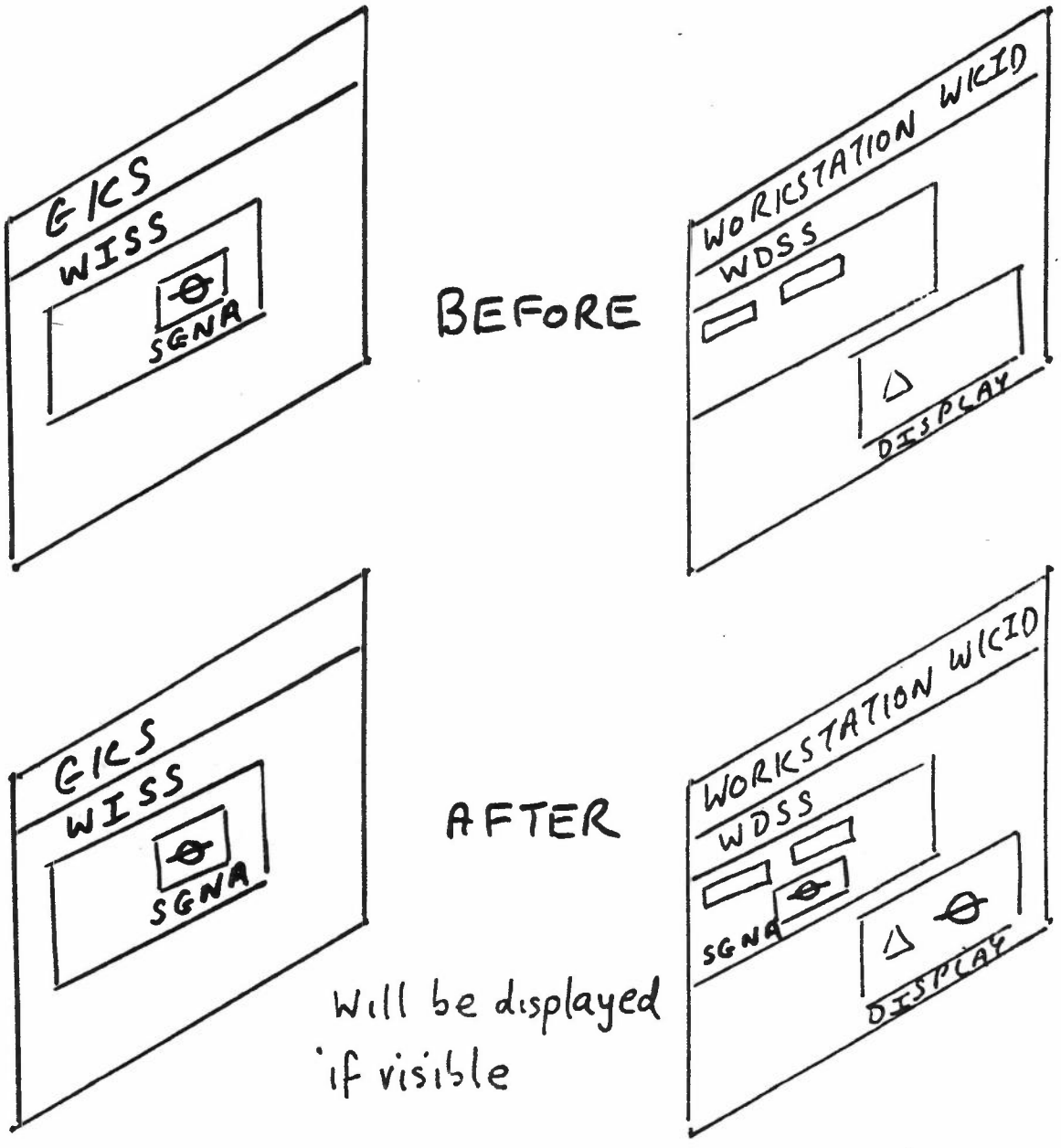
WISS

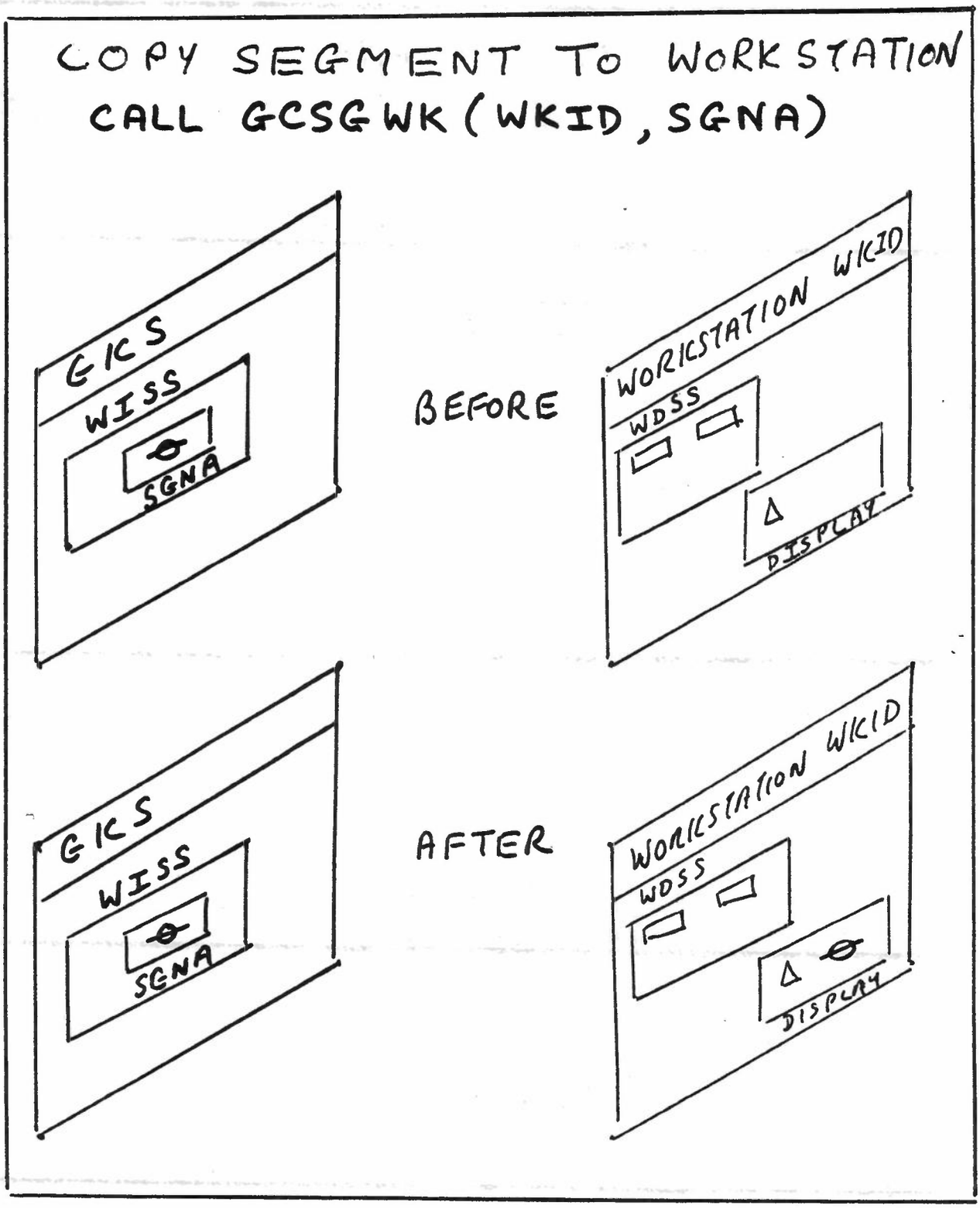
GKS defines a virtual workstation with its own segment storage

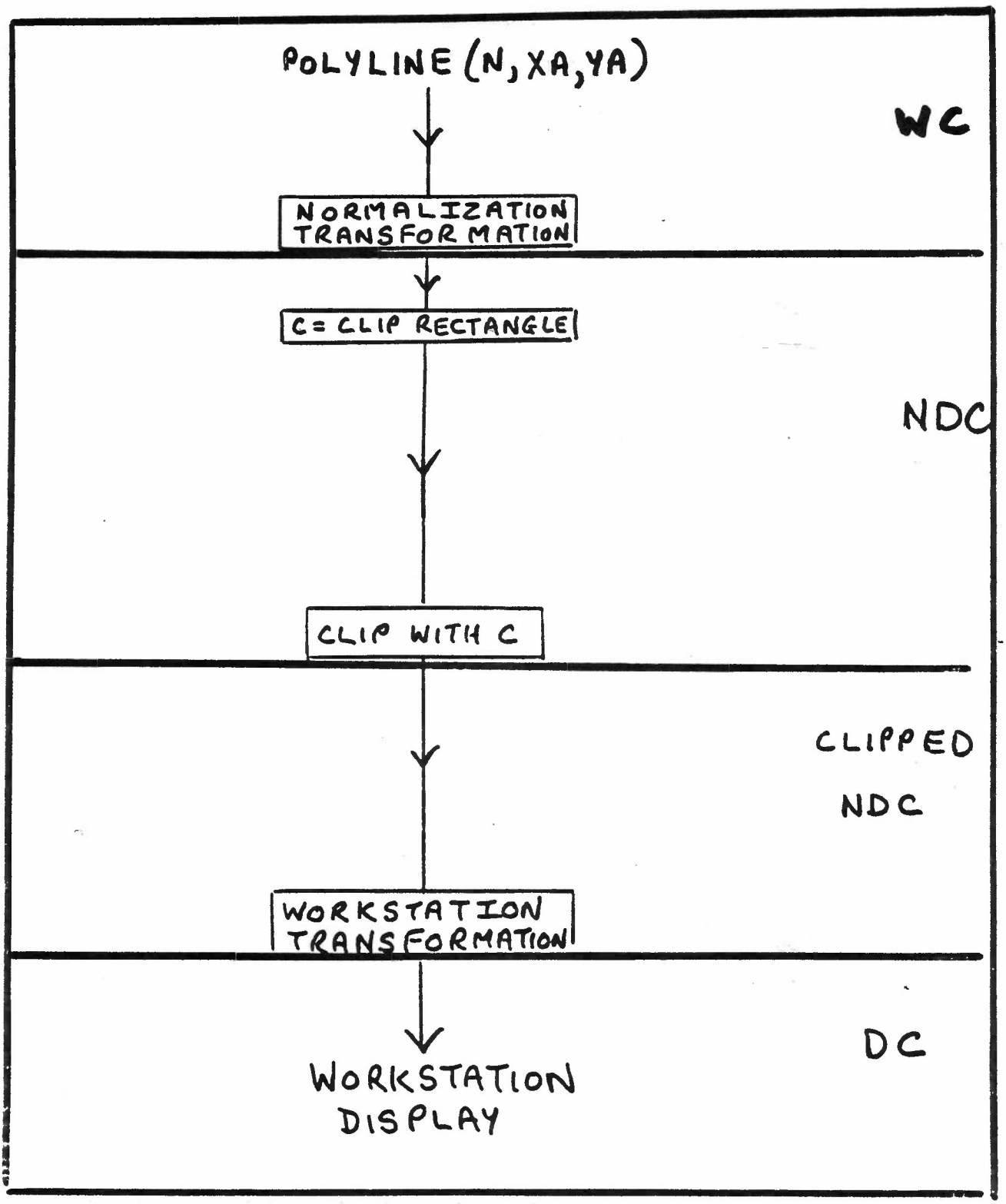


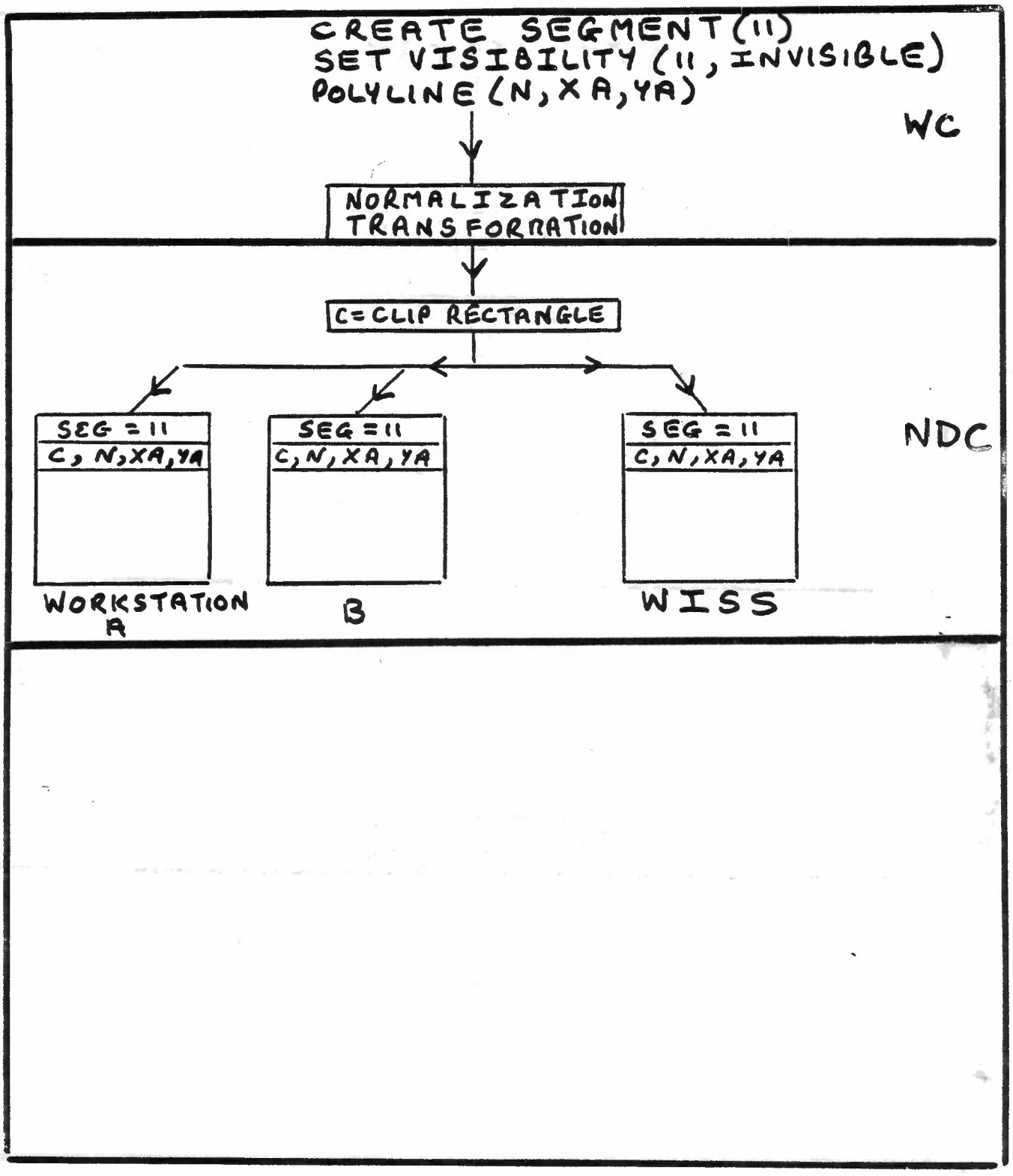
ASSOCIATE SEGMENT WITH WORKSTATION

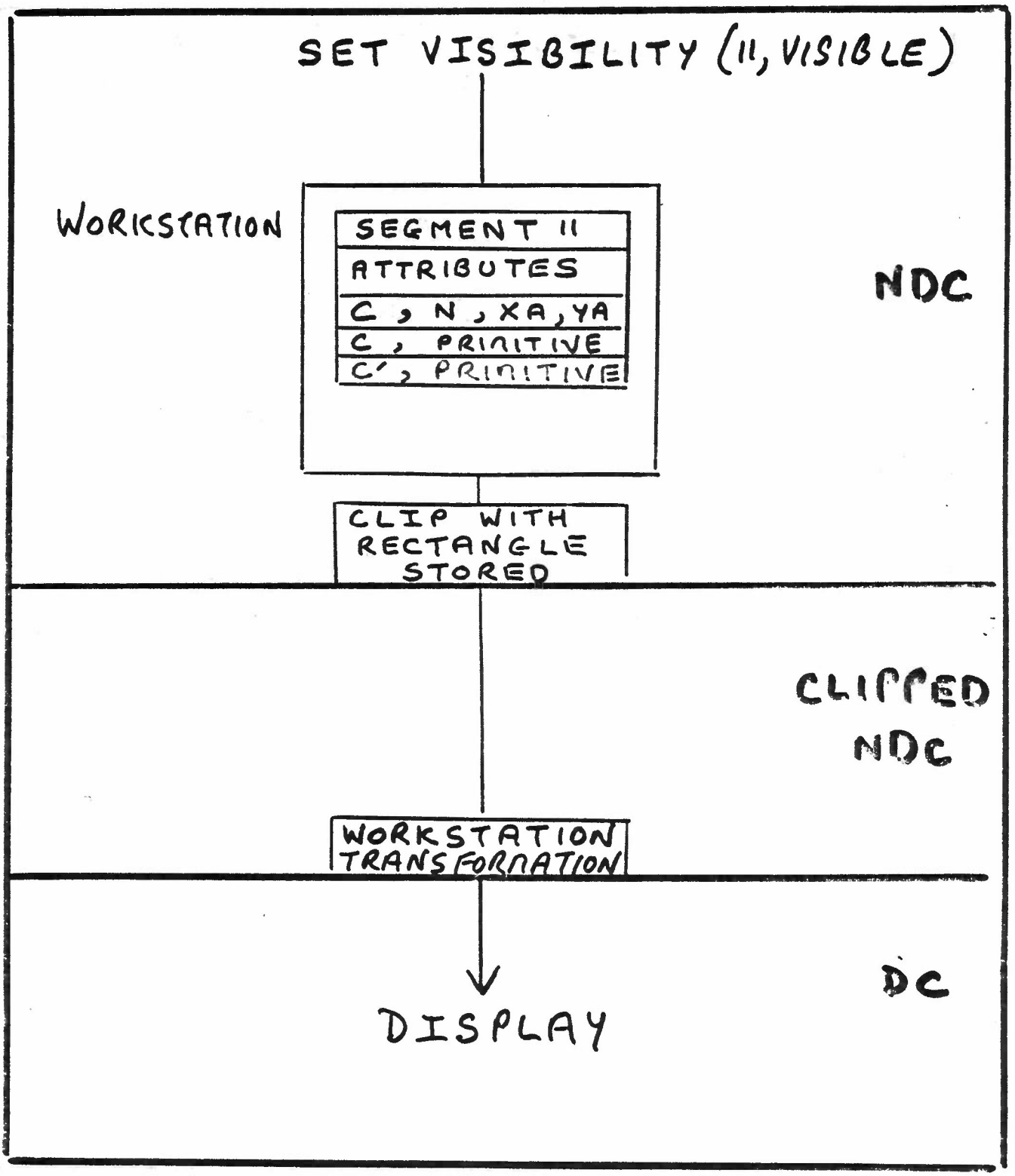
CALL GASGWK (WKID, SGNA)

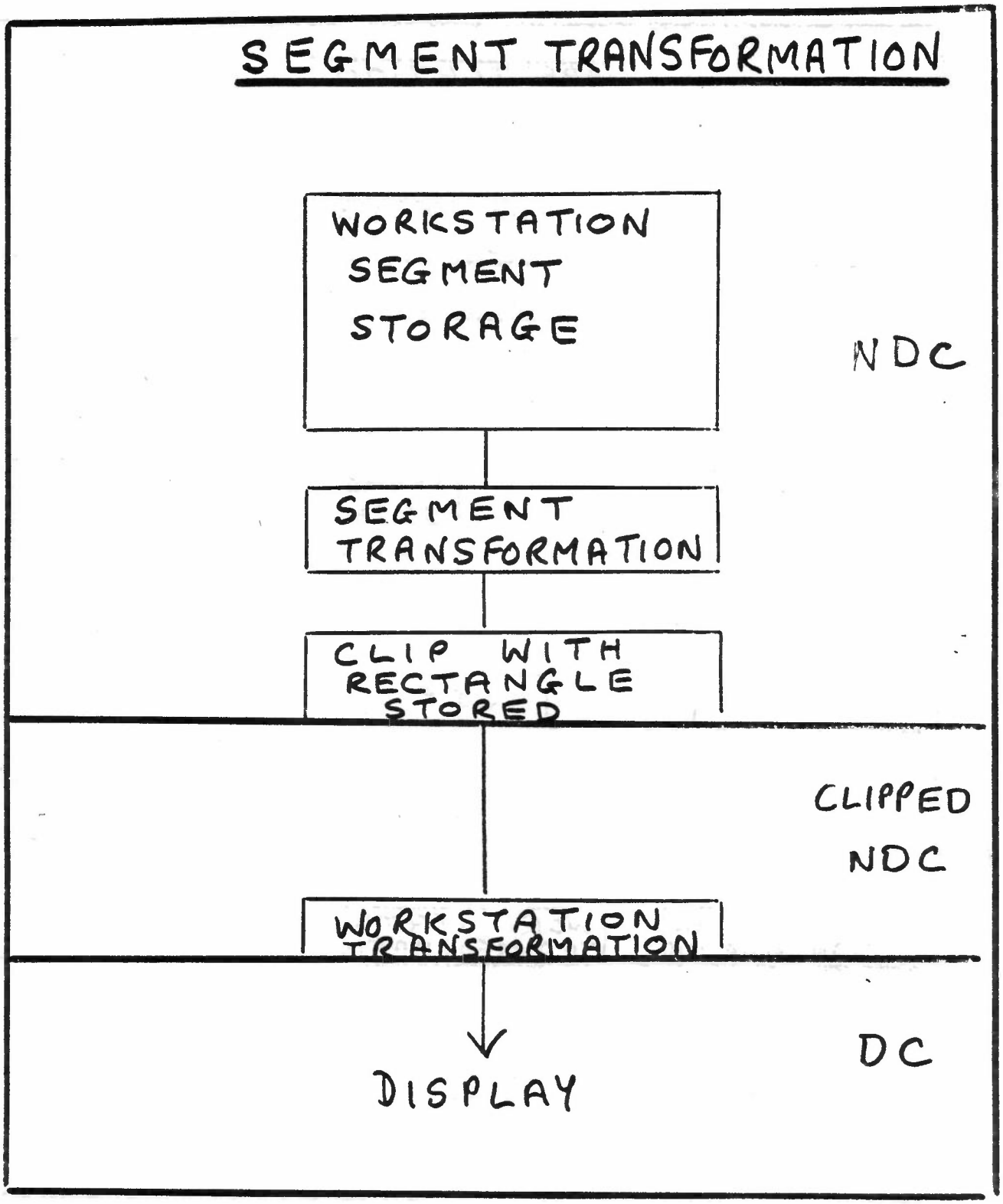


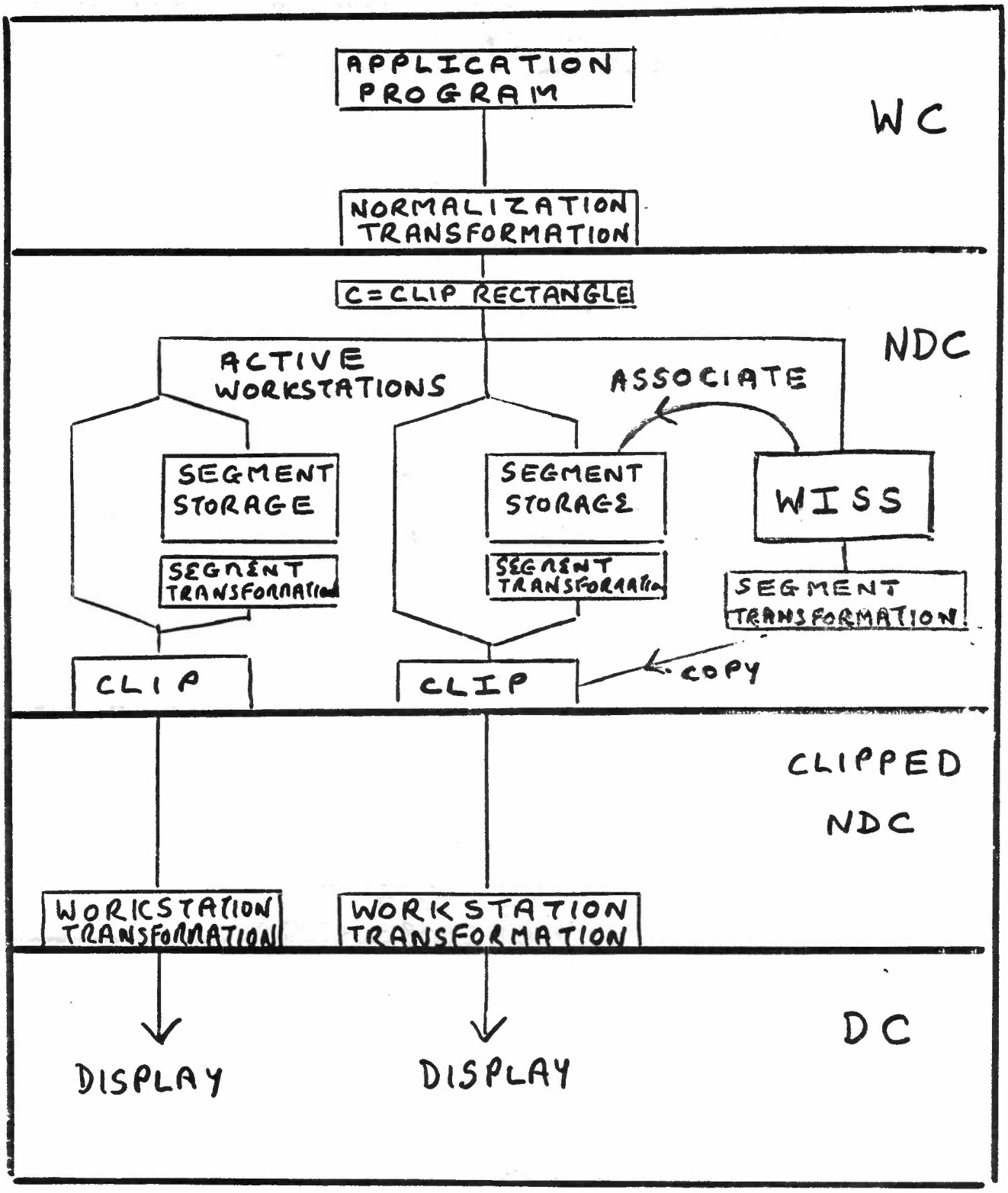

COPY SEGMENT TO WORKSTATION
CALL GCSGWK (WKID, SGNA)
GICS
WISS
SGNA
BEFORE
WORKSTATION WKID
WDSS
DISPLAY
GICS
WISS
SGNA
AFTER
WORKSTATION WKID
WDSS
DISPLAY


POLYLINE (N, XA, YA)
WC
NORMALIZATION TRANSFORMATION
C = CLIP RECTANGLE
NDC
CLIP WITH C
CLIPPED NDC
WORKSTATION TRANSFORMATION
DC
WORKSTATION DISPLAY


CREATE SEGMENT (11)
SET VISIBILITY (11, INVISIBLE)
POLYLINE (N, XA, YA)
WC
NORMALIZATION TRANSFORMATION
C=CLIP RECTANGLE
SEG = 11
C, N, XA, YA
SEG = 11
C, N, XA, YA
SEG = 11
C, N, XA, YA
NDC
WORKSTATION A
B
WISS


SET VISIBILITY (11, VISIBLE)
WORKSTATION
SEGMENT 11
ATTRIBUTES
C , N , XA , YA
C , PRIMITIVE
C', PRIMITIVE
NDC
CLIP WITH RECTANGLE STORED
CLIPPED NDC
WORKSTATION TRANSFORMATION
DC
DISPLAY


SEGMENT TRANSFORMATION
WORKSTATION SEGMENT STORAGE
NDC
SEGMENT TRANSFORMATION
CLIP WITH RECTANGLE STORED
CLIPPED NDC
WORKSTATION TRANSFORMATION
DC
DISPLAY

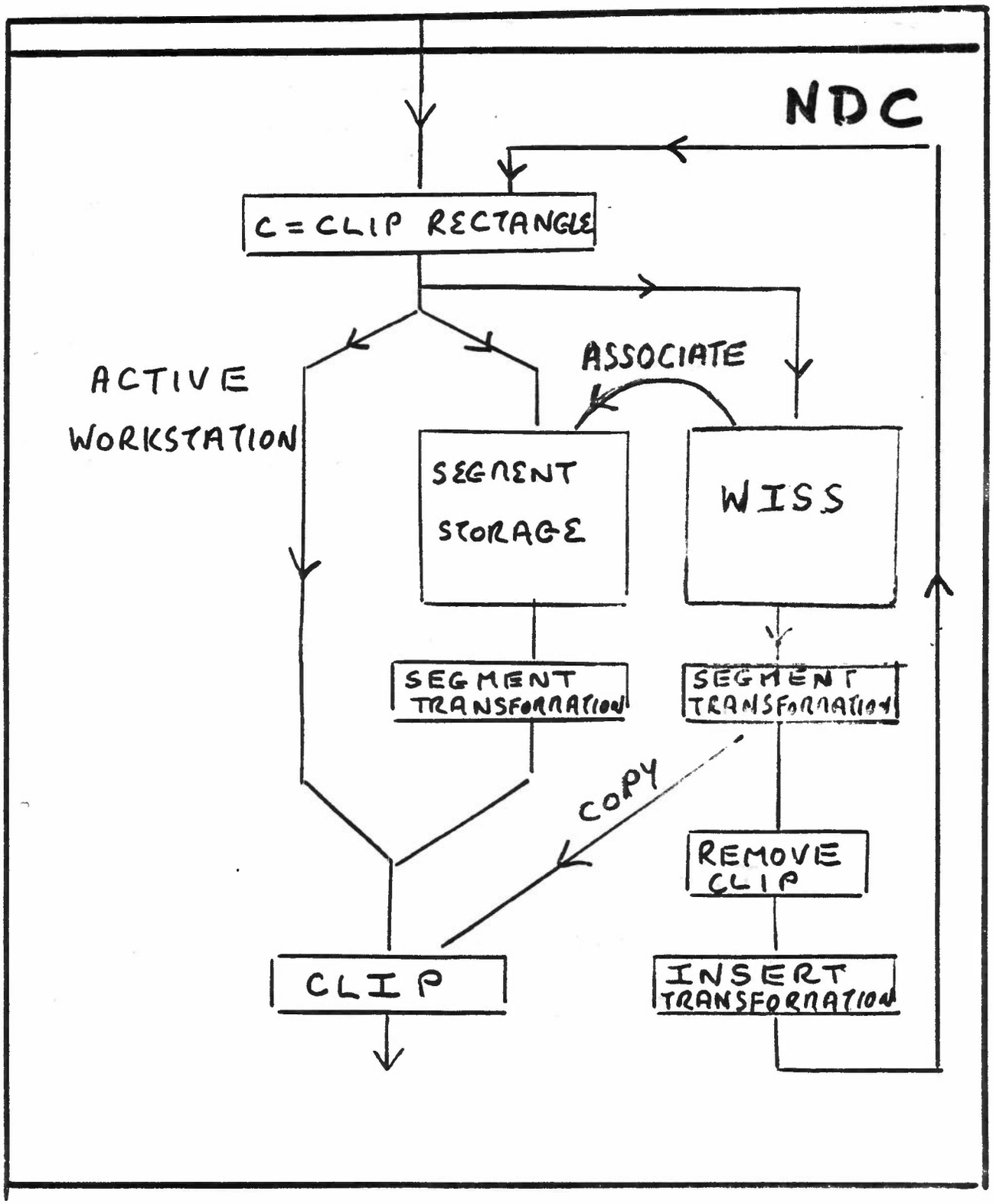

APPLICATION PROGRAM
WC
NORMALIZATION TRANSFORMATION
C=CLIP RECTANGLE
NDC
ACTIVE WORKSTATIONS
ASSOCIATE
SEGMENT STORAGE
SEGMENT STORAGE
WISS
SEGMENT TRANSFORMATION
SEGMENT TRANSFORMATION
SEGMENT TRANSFORMATION
CLIP
CLIP
COPY
CLIPPED NDC
WORKSTATION TRANSFORMATION
WORKSTATION TRANSFORMATION
DC
DISPLAY
DISPLAY

INSERT SEGMENT

CALL GINSG(SEG, MAT)

Copies the primitives in segment SEG to all the active workstations.

- Apply segment transformation
- then apply insert transformation specified by MAT
- If segment open, primitives become part of that segment
- The clipping rectangle stored with the primitives in SEG is ignored. The current one is used


NDC
C = CLIP RECTANGLE
ACTIVE WORKSTATION
ASSOCIATE
SEGMENT STORAGE
WISS
SEGMENT TRANSFORMATION
SEGMENT TRANSFORMATION
COPY
REMOVE CLIP
CLIP
INSERT TRANSFORMATION

```
OPEN WORKSTATION (SS, 5, 1)
OPEN WORKSTATION (DISPLAY, 6, 4)

SET WINDOW (1, 0, 70, 0, 70)
SET VIEWPORT (1, 0, 0·5, 0·5, 1)
SET WINDOW (2, 0, 30, 0, 30)
SET VIEWPORT (2, 0, 0·5, 0, 0·5)

ACTIVATE WORKSTATION (SS)
CREATE SEGMENT (HS)
SELECT NORMALIZATION TRANSFORMATION (1)
HOUSE
CLOSE SEGMENT

CREATE SEGMENT (BT)
SELECT NORMALIZATION TRANSFORMATION (2)
BOAT
CLOSE SEGMENT
```

WISS contains segments
HS and BT

ASSOCIATE SEGMENT WITH
WORKSTATION (DISPLAY, HS)

Segment HS will be stored on workstation as if DISPLAY had been active when segment defined

- Note DISPLAY need not be active

COPY SEGMENT TO WORKSTATION
(DISPLAY, BT)

Picture on workstation display would be the same as if it had been active when BT defined BUT segment not stored on workstation DISPLAY

```
ACTIVATE WORKSTATION (DISPLAY)
EVALUATE TRANSFORMATION MATRIX
     (0, 0, 0, 0, 0, 1, 1, WC, IDEN)

CREATE SEGMENT (NEW)
INSERT SEGMENT (HS, IDEN)
INSERT SEGMENT (BT, IDEN)
CLOSE SEGMENT
```

Segment NEW containing both HOUSE and BOAT will be defined on active workstations

CLIPPING

```
SET CLIPPING INDICATOR (CLIP)
EVALUATE TRANSFORMATION MATRIX
     (0, 0, 0, 0·25, 0, 1, 1, NDC, MAT)

ACTIVATE WORKSTATION (SS)
CREATE SEGMENT (HS)
SET SEGMENT TRANSFORMATION (HS, MAT)
SELECT NORMALIZATION TRANSFORMATION (1)
HOUSE
CLOSE SEGMENT

EVALUATE TRANSFORMATION MATRIX
     (0, 0, 0·25, 0, 0, 1, 1, NDC, MAT2)
CREATE SEGMENT (BT)
SET SEGMENT TRANSFORMATION (BT, MAT2)
SELECT NORMALIZATION TRANSFORMATION (2)
BOAT
CLOSE SEGMENT
```

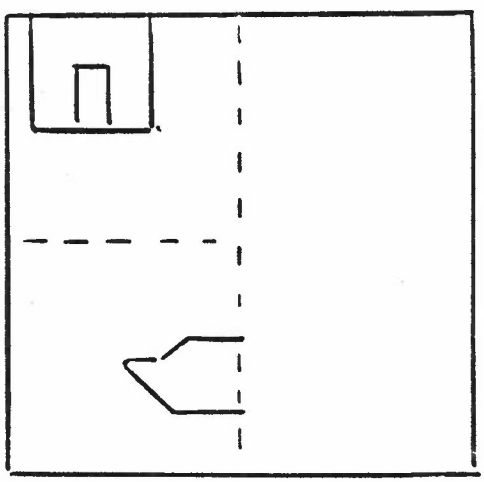
SEGMENT HS

Segment Transformation ↑ 0.25 MAT
Clipping Rectangle $x = 0 \rightarrow 0.5$
HOUSE $y = 0.5 \rightarrow 1$

SEGMENT BT

Segment Transformation → 0.25 MAT2
Clipping Rectangle $x = 0 \rightarrow 0.5$
BOAT $y = 0 \rightarrow 0.5$

ACTIVATE WORKSTATION (DISPLAY)
ASSOCIATE SEGMENT WITH WORKSTATION (DISPLAY, HS)
ASSOCIATE SEGMENT WITH WORKSTATION (DISPLAY, BT)



```
ACTIVATE WORKSTATION (DISPLAY)
SET VIEWPORT (3, 0, 0·75, 0, 0·75)
SET CLIPPING INDICATOR (CL)
EVALUATE TRANSFORMATION MATRIX
    (0, 0, 0, -0·125, 0, 1, 1, NDC, MAT3)
CREATE SEGMENT (NEW)
SELECT NORMALIZATION TRANSFORMATION (3)
INSERT SEGMENT (HS, MAT3)
INSERT SEGMENT (BT, MAT3)
CLOSE SEGMENT
```

SEGMENT NEW

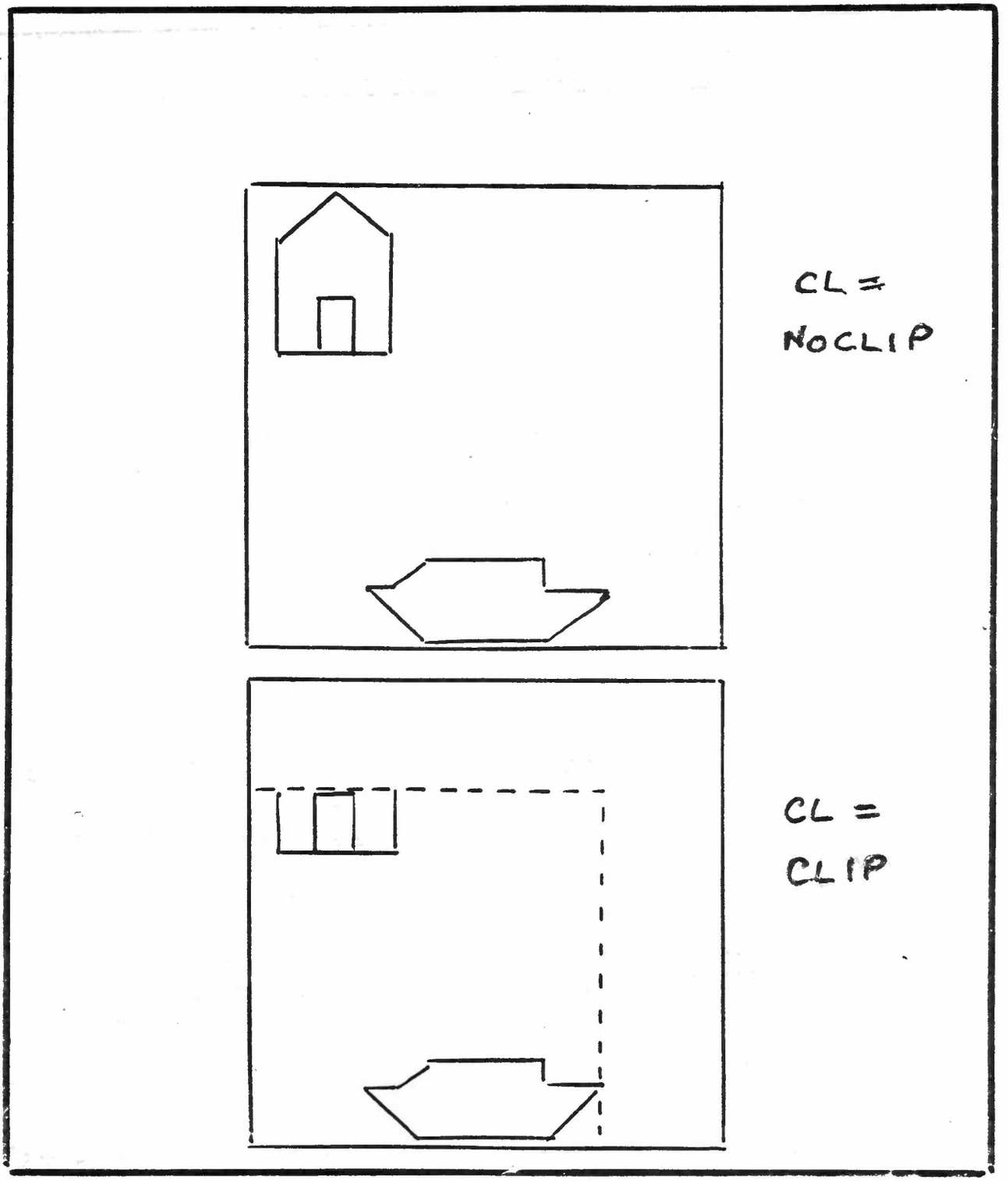
Segment Transformation = Identity

Clipping Rectangle = (0, 1, 0, 1) CL = NOCLIP

(0, 0·75, 0, 0·75) CL = CLIP

HOUSE ↑0·25 ↓0·125

BOAT →0·25 ↓0·125


CL =
NOCLIP
CL =
CLIP

DEFERRING PICTURE CHANGES

It may be inefficient to update workstation immediately there is a picture change

SET DEFERRAL STATE (WSID, DEFERRAL MODE, IMPLICIT REGENERATION MODE)
CALL GSDS (WKID, DEFMOD, REGMOD)

DEFERRAL MODE

Allows workstation to delay output.

ASAP	As Soon as Possible
BNIG	Before Next Interaction Globally
BNIL	Before Next Interaction Locally
ASTI	At Some TIme

IMPLICIT REGENERATION

Certain function cause picture to be regenerated

SUPPRESSED	implicit regeneration suppressed
ALLOWED	implicit regeneration allowed

DEFERRAL MODE EXAMPLES

- Off-line plotter output spooled to magnetic tape **ASTI**
- Storage Tube connected to host via network **BNIL**
- Operator using two storage tubes and inputting from both sets of thumbwheels **BNIG**
- Highly interactive vector refresh display **ASAP**

IMPLICIT REGENERATION

Following functions may imply an implicit regeneration

- SET POLYLINE REPRESENTATION etc
- SET PATTERN REPRESENTATION
- SET COLOUR REPRESENTATION
- Change WORKSTATION window or viewport
- Change segment attribute
- Segment deletion
- Output of primitives when segment priority is important

Workstation specifies which it can do immediately

UPDATE WORKSTATION

CALL GUWK (WS, RF)

Forces the display to be current
All deferred actions are performed

If RF = PERFORM, an implicit regeneration is done if it is needed.

If RF = POSTPONE, no implicit regeneration done

UPDATE WORKSTATION (WS, PERFORM)
similar effect to:

SET DEFERRAL STATE (ASAP, ALLOWED)

without changing deferral state

CLEAR WORKSTATION

CALL GCLRWK (WKID, COFL)

COFL (control flag) = CONDITIONALLY
ALWAYS

Deletes current picture ready for a new image to be output

- All deferred actions executed
- If display surface is already clear, will only force another clear if COFL = ALWAYS (allows blank frames)
- Updates workstation window/viewport
- Deletes all segments on workstation
- Display surface is noted as 'EMPTY'

REDRAW ALL SEGMENTS ON WORKSTATION

REDRAW ALL SEGMENTS ON WORKSTATION

```
CALL GRSGWK (WKID)
```

All deferred actions for workstation WKID are executed,

The display surface is cleared,

All visible segments stored on the workstation are redisplayed

GKS ENVIRONMENT

OPEN GKS

```
CALL GCPKS (EF, BUFA)
```

Opens GKS. EF is the name of file used by GKS to return error information

Initialises many GKS variables to default values. For example:

- max number simultaneous open workstations
- default linestyles
- default index settings

BUFA is amount of memory units - implementation dependent (-1 uses implementation default)

CLOSE GKS

```
CALL GCLKS
```

Flushes buffers, closes error file etc

GKS STATE LIST

Contains state variables and default information of global significance
Some of the entries :

- Set of open workstations
- Set of active workstations
- current normalization transformation
- viewport input priority list
- windows and viewports
- current primitive index values
- current aspects
- current pattern size and ref point
- current pick identifier
- name of open segment
- set of segment names in use
- event queue
- . . . .
- . . . .

GKS OPERATING STATES

To aid error handling, GKS defines a set of operating states. Specific GKS functions allowed in each state

GKS OPEN

WORKSTATION OPEN

WORKSTATION ACTIVE

SEGMENT OPEN

Strict ordering to states

WORKSTATION DESCRIPTION TABLE

Defines the types of workstation available at the installation. CONTAINS:

- Workstation category (OUTPUT, OUTIN ..)
- Device coordinates
- Display surface size
- Raster or vector
- Dynamic modification status for representations of primitives
- Default deferral/implicit regeneration mode
- Linetypes etc available (same for other primitives)
- Support for segment priority
- Dynamic modification status for segments
- Defaults for logical input devices

WORKSTATION STATE LIST

Contains information specific to a particular open workstation. Defaults taken from workstation description table.

- workstation identifier
- connection identifier
- workstation type
- workstation state (ACTIVE or not)
- set of stored segments
- deferral mode / implicit regeneration
- display surface empty
- new frame action necessary at update
- primitive bundle tables
- current & requested workstation window
- " viewport
- logical input device information
- etc

INQUIRY FUNCTIONS

Allow application programmer to access information in the State Lists

- Aids error diagnostic
- Library Routines
- Tune application

Typical form :

INQUIRE ____ (IND , info)

- IND = 0 , info is valid
- IND ≠ 0 , contains error number

Not possible to get a GKS error when calling INQUIRE

INQUIRE DEFAULT DEFERRAL STATE
VALUES (WST, IND, DFMODE, IMPMODE)

IND = 0 then
DFMODE set to ASAP, BNIG etc
IMPMODE set to SUPPRESSED or ALLOWED

IND = 8 GKS not in proper state (GKS not OPEN)

= 22 Specified workstation type WST is invalid

= 23 Specified workstation type WST does not exist

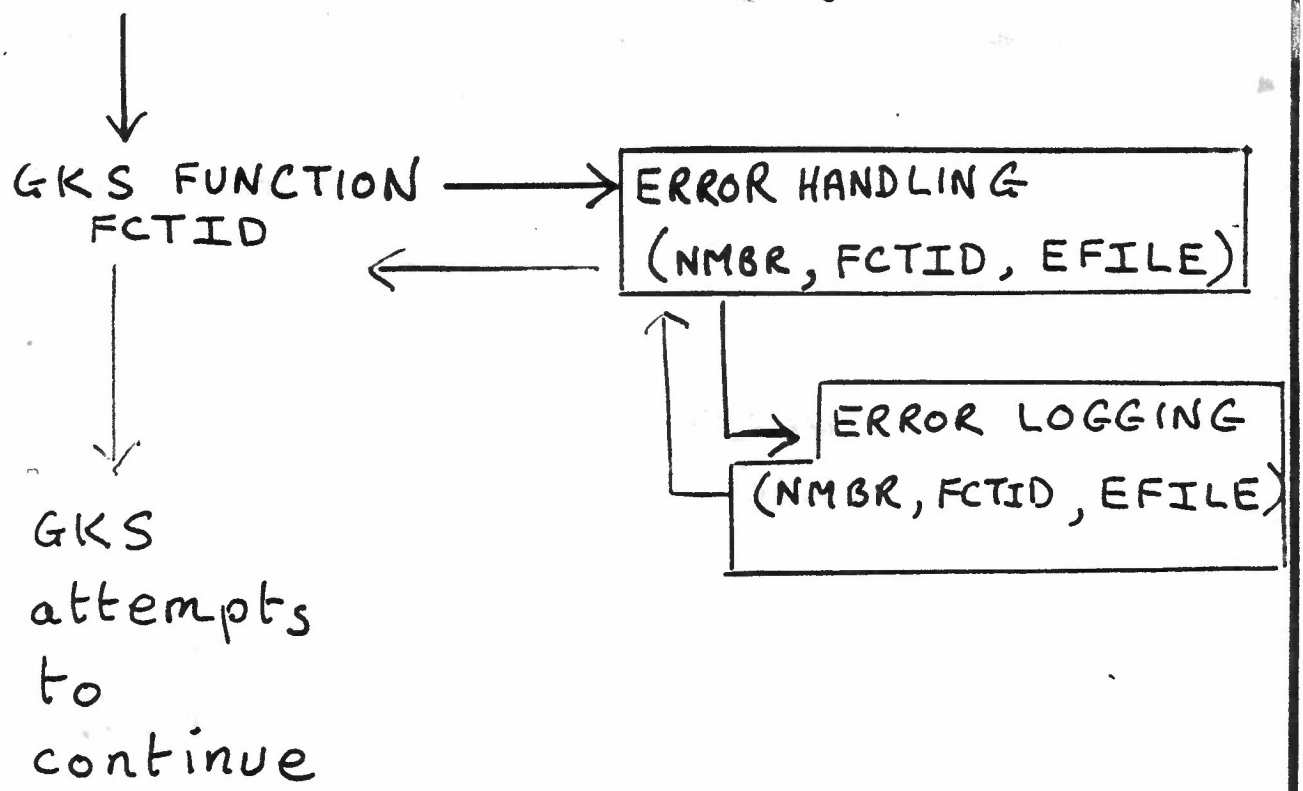
= 29 Specified workstation type WST is not OUTPUT or OUTIN workstation

ERROR HANDLING

All errors reported by entry in error file :

(1) Error Number

(2) GKS Function being executed

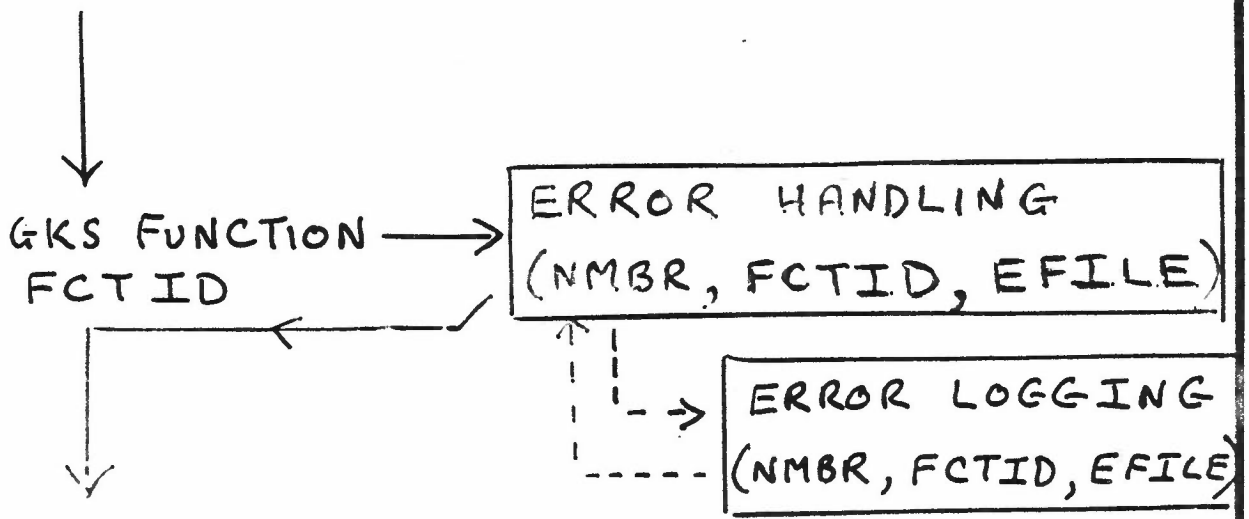


GKS ERRORS

1 → 8	Incorrect States
20 → 41	Workstation does not exist, not open/active, wrong type etc
50 → 54	Wrong transformation number, rectangle definition invalid, workstation window not in NDC unit square etc
60 → 89	Output attributes wrong – index, type, etc
100 → 104	Number of points wrong in output primitive, invalid GDP etc
120 → 126	Segment errors

etc

USER SUPPLIED ERROR HANDLING



- User supplied ERROR HANDLING routine can provide actions tailored to the application.
- Recommended that it still calls ERROR LOGGING

```
OPEN GKS (EFILE, -1)
OPEN WORKSTATION (1, 6, 3)
ACTIVATE WORKSTATION (1)
CLOSE SEGMENT
```

- Error generated - SEGMENT not open - Error Number 4

USER SUPPLIED ERROR HANDLING

```
SUBROUTINE ERROR HANDLING
      (NMBR, FCTID, EFILE)
IF ( NMBR .NE. 4) GOTO 100
```

Use INQUIRY functions to see which Workstations were active perhaps

```
100  ERROR LOGGING (NMBR, FCTID, EFILE)
     RETURN
     END
```

EMERGENCY CLOSE GKS

CALL GECLKS

Implementation dependent

May be called by application or GKS

- Tries to close workstations

GENERALIZED DRAWING PRIMITIVE

GDP

CALL GDP (N, PX, PY, PRIMID, IL, IA)

PRIMID, the GDP identifier, defines which type of GDP to draw

N, PX, PY define a set of points used by GDP. These are defined in world coordinates and will get transformed just like any other point

IA is a data record IA(1) → IA(L) providing additional non-geometric information

GENERALIZED DRAWING PRIMITIVE

- Any primitive attributes can be bound to the GDP
- GDPs are subject to registration
- GDPs must be clipped correctly
- Allows access to hardware features
  - circle
  - arc
  - ellipse
  - Bezier curve
- Geometry should be maintained
  Circles become ellipses under differential scaling

MESSAGE

CALL GMSG (WKID, MESS)

Character string sent to Operator

Application program has no control on how it gets displayed (installation could use different device)

CHANGE PAPER ON PLOTTER

MOUNT CINE CAMERA

PUT OUTPUT ON NEW MAGNETIC TAPE (to METAFILE workstation)

PLUG IN TABLET

ESCAPE

CALL GESC (FCTID, LIDR, IDR, MLODR, LODR, ODR)

FCTID is function identification

LIDR length of input data record IDR

MLODR max length of output data record

LODR length of output data record ODR

YOU CAN DO ANYTHING AS LONG AS:

a) GKS design concept not violated.

b) GKS state lists not altered.

c) No geometric output

d) Side effects well documented.

- Multiple display buffers
- RASTER OP
- BELL – VOICE I/O !

GKS LOGICAL INPUT DEVICES

GKS defines six classes :

LOGICAL INPUT DEVICE	PROVIDES
LOCATOR	• position in world coordinates and normalization transformation number
VALUATOR	• a real number
CHOICE	• a non-negative integer representing a selection from a set of choices
PICK	• a segment name and pick identifier
STRING	• a character string
STROKE	• sequence of positions in world coordinates and normalization transformation number

Device can also return status information

PICK IDENTIFIER

- PICK logical input device used to identify part of picture of interest
- Major identification via SEGMENT NAME
- PICK IDENTIFIER provides a second level of naming of primitives within a segment

REQUEST PICK

CALL GRQPK(WS, DV, ST, SEG, PICKID)

Returns the name of a segment SEG and a more specific identification within segment by PICKID

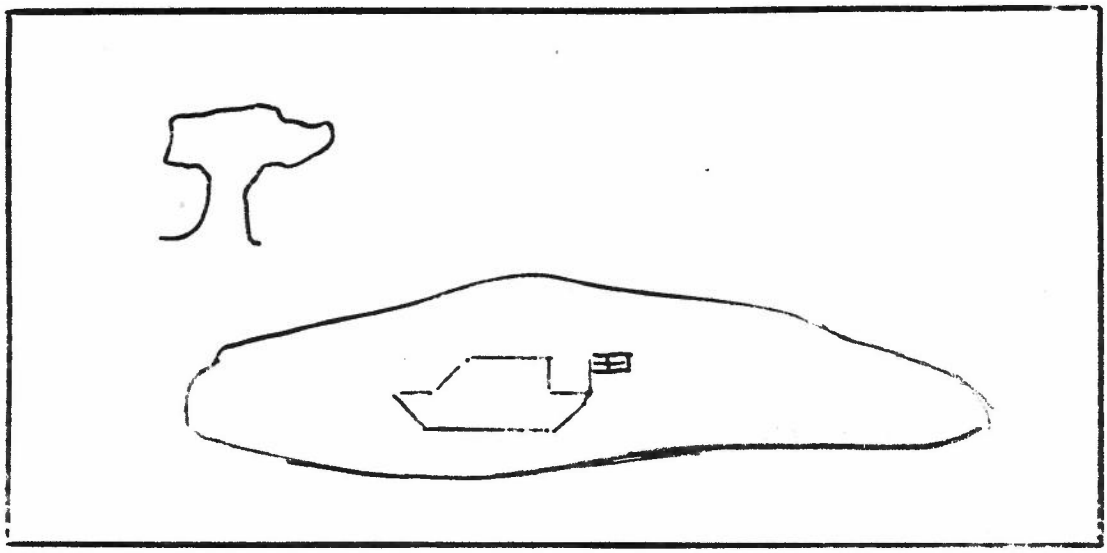
SET PICK IDENTIFIER

CALL GSPKID (N)

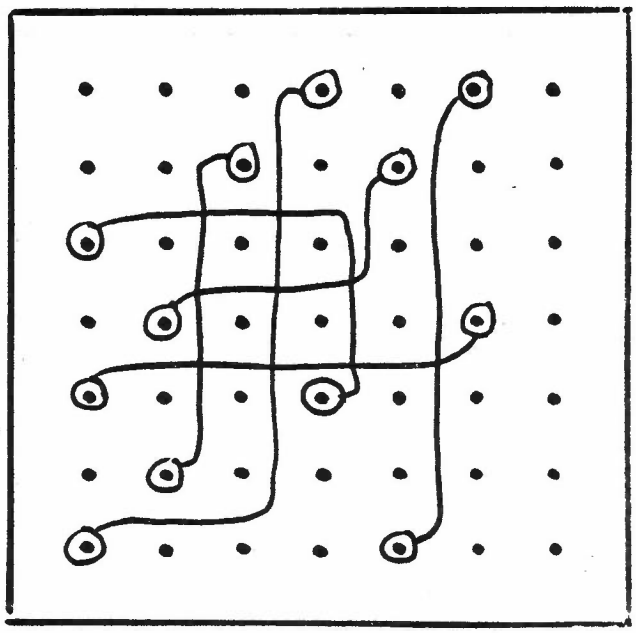
Subsequent output primitives within segments will have a pick identifier value of N associated with them until another call of SET PICK IDENTIFIER

Primitives outside segments do not have an associated pick identifier

	SEGMENT NAME	PICK IDENTIFIER
POLYLINE (NA, XA, YA)	–	–
SET PICK IDENTIFIER (3)		
POLYLINE (NB, XB, YB)	–	–
CREATE SEGMENT (10)		
POLYLINE (NC, XC, YC)	10	3
POLYLINE (ND, XD, YD)	10	3
SET PICK IDENTIFIER (4)		
POLYLINE (NE, XE, YE)	10	4
POLYLINE (NF, XF, YF)	10	4
CLOSE SEGMENT		
POLYLINE (NG, XG, YG)	–	–
CREATE SEGMENT (11)		
POLYLINE (NH, XH, YH)	11	4
SET PICK IDENTIFIER (3)		
POLYLINE (NI, XI, YI)	11	3
SET PICK IDENTIFIER (4)		
POLYLINE (NJ, XJ, YJ)	11	4
CLOSE SEGMENT		



```
      CREATE SEGMENT (1)
      TREE
      CLOSE SEGMENT

      CREATE SEGMENT (2)
      BOAT
      CLOSE SEGMENT

      CREATE SEGMENT (3)
      POND
      CLOSE SEGMENT

100   CONTINUE

      REQUEST PICK (WS, DV1, ST1, SEG, PICKID)

      REQUEST LOCATOR (WS, DV2, ST2, NT, X, Y)

      EVALUATE TRANSFORMATION MATRIX (0,0,X,Y,0,1,1, NDC, MAT)

      SET SEGMENT TRANSFORMATION (SEG, MAT)

      GOTO 100
```



```
      CREATE SEGMENT (1)
      DO 100 J=1,JMAX
      SET PICK IDENTIFIER (J)
      DRAW CONNECTION (J)
100   CONTINUE
      CLOSE SEGMENT

      REQUEST PICK (WS,DV,ST,SEG,PICKID)

      returns  SEG = 1
               PICKID = J
```

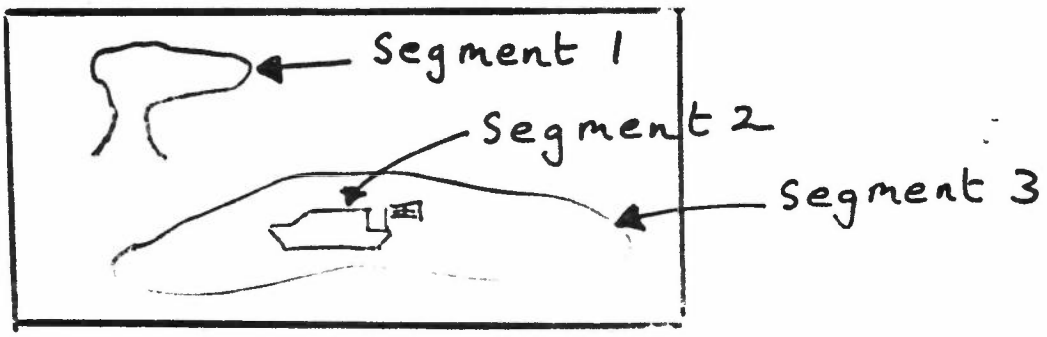
SEGMENT DETECTABILITY

Segment attribute SET DETECTABILITY

CALL GSDTEC (SEG, DET)

Sets detectability of segment SEG to DET, either DETECTABLE or UNDETECTABLE

Default is UNDETECTABLE



SET DETECTABILITY (1, DETECTABLE)
SET DETECTABILITY (2, DETECTABLE)
SET DETECTABILITY (3, UNDETECTABLE)

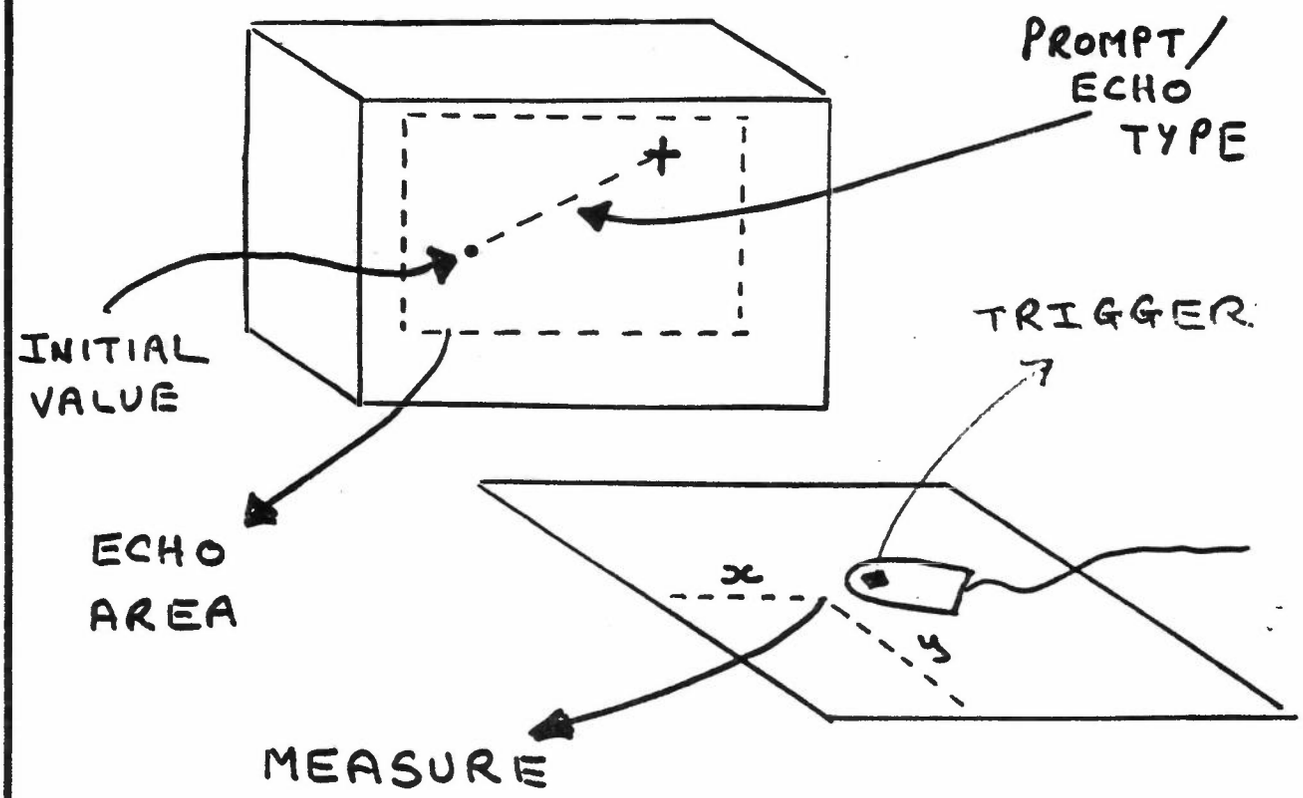
REQUEST PICK(WS, DV, ST, SEG, PICKID)

INPUT MODES

Each Logical Input Device can be operated in three modes:

REQUEST	reads a logical input value from specified Logical Input Device. GKS waits until input is entered by operator or break occurs
SAMPLE	GKS without waiting for operator returns current logical input value of specified L.I.D
EVENT	GKS maintains an input queue of temporally ordered event reports which give name of device and logical input value. Event reports are generated asynchronously by operator

GKS LOGICAL INPUT DEVICE MODEL



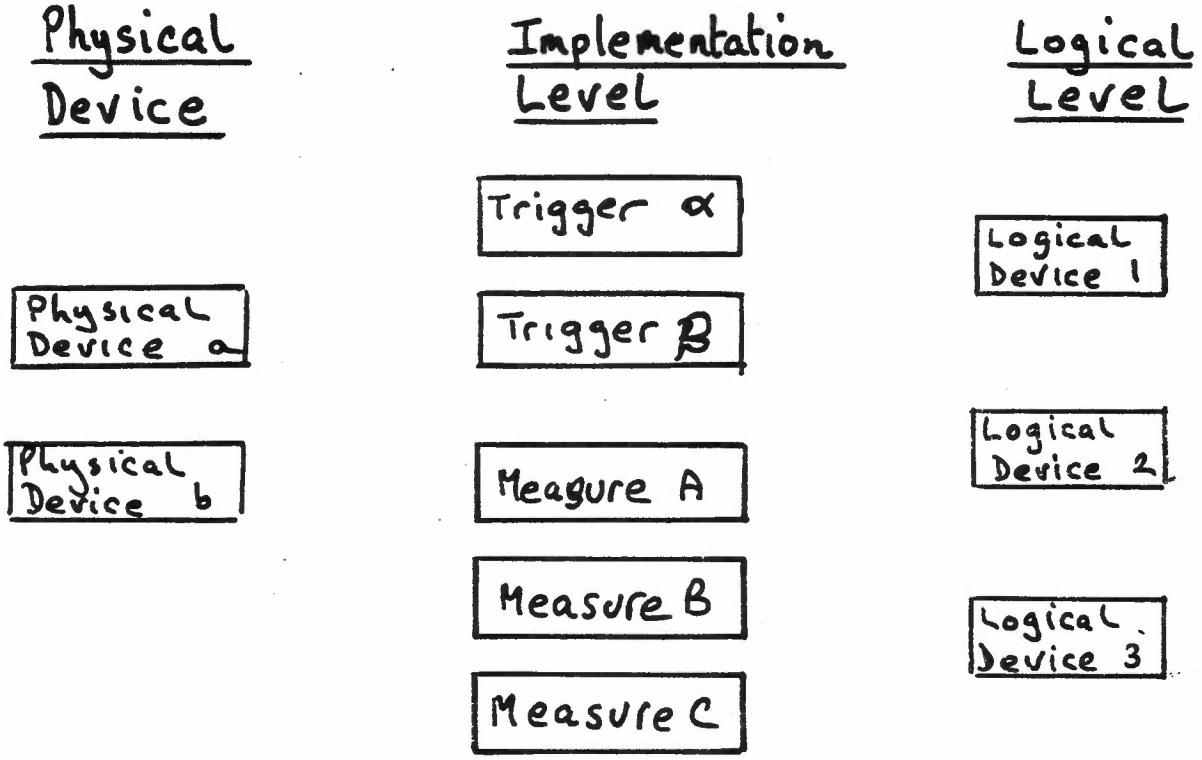
MEASURE value determined by one or more physical input devices. Can be thought of as a process

TRIGGER independent process that sends a message to one or more Logical Input Devices

LOGICAL INPUT DEVICE MODEL

MEASURE : value determined by one or more devices.

TRIGGER : independent process sending messages to one or more logical input devices



GKS LOGICAL INPUT DEVICE MODEL

PROMPT/ECHO TYPE

Defines form of prompting and echoing to be used

INITIAL VALUE

Supplied by programmer

ECHO AREA

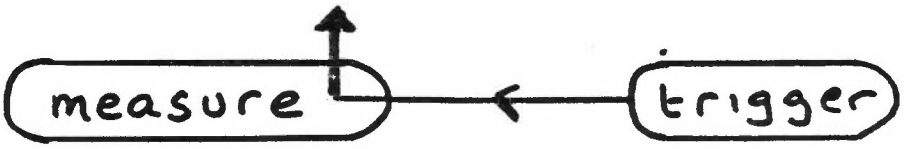
Indicates where echoing should take place

DATA RECORD

Additional information relevant to a specific device

GKS LOGICAL INPUT DEVICE MODEL

REQUEST MODE



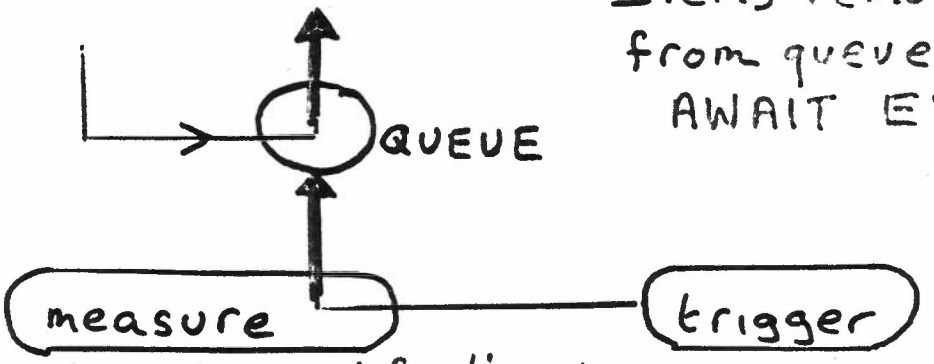
Single value returned to application when trigger fires

SAMPLE MODE

Trigger not used. Value returned for each SAMPLE

EVENT MODE

Items removed from queue by AWAIT EVENT



Value and device identification to single queue when trigger fires

MODE SETTING

Each device can be in only one of the modes REQUEST, SAMPLE and EVENT

SET LOCATOR MODE (WS, DV, MODE, EC)

MODE = REQUEST, SAMPLE or EVENT
EC = ECHO or NOECHO

DEFAULT MODE IS REQUEST

DEFAULT EC IS ECHO

SET LOCATOR MODE

```
CALL GSLCM (WS, DV, MODE, EC)
```

SET VALUATOR MODE

```
CALL GSVLM (WS, DV, MODE, EC)
```

SET CHOICE MODE

```
CALL GSCHM (WS, DV, MODE, EC)
```

SET PICK MODE

```
CALL GSPKM (WS, DV, MODE, EC)
```

SET STRING MODE

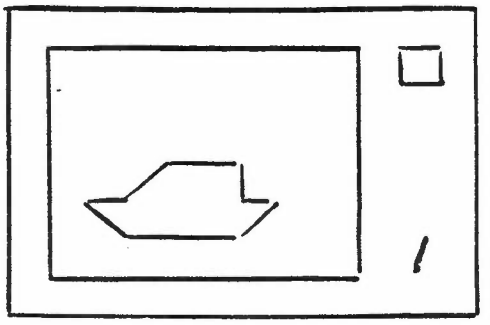
```
CALL GSSTM (WS, DV, MODE, EC)
```

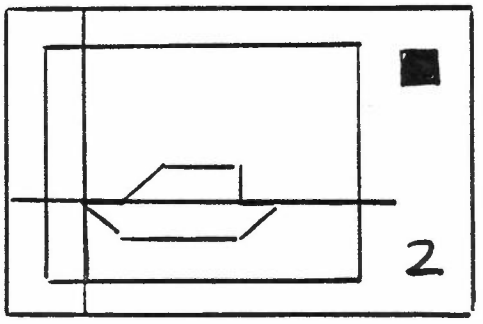
SET STROKE MODE

```
CALL GSSKM (WS, DV, MODE, EC)
```

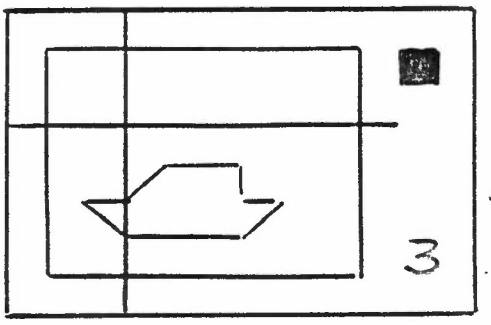
REQUEST LOCATOR (WS, DV, ST, N, X, Y)
REDRAW BOAT AT (X, Y)

MEASURE process started
ECHOING started
TRIGGER process started
GKS is stopped

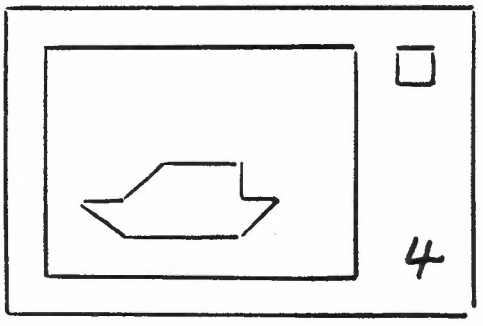




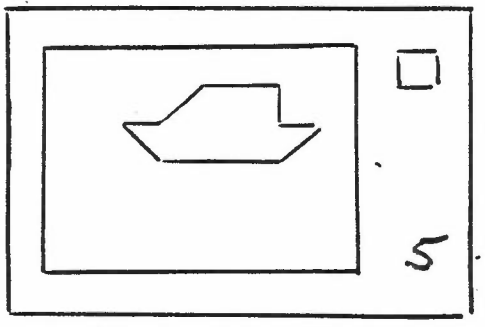
Operator moves cross hairs →



Presses button ←



REQUEST LOCATOR complete



BOAT redrawn

STATUS

STATUS parameter is returned to indicate success or failure

OK	success
NONE	failure

NONE indicates input data invalid
Operator has indicated a break
(invalid locator position, separate trigger)

```
100   CONTINUE
      REQUEST LOCATOR (WS, DV, ST, N, X, Y)
      IF (ST .EQ. NONE) GOTO 200
      REDRAW BOAT AT (X, Y)
      GOTO 100
200   CONTINUE
```

PICK STATUS

PICK device has an additional status that can be returned

OK	success
NONE	failure
NOPICK	invalid PICK (PICK device not pointing at segment)

CHOICE STATUS

CHOICE device has an additional status that can be returned

OK	success
NONE	failure
NOCHOICE	no choice has been made

SAMPLE MODE

SET LOCATOR MODE(WS, DV, SAMPLE, EC)

- Mode of specified device set to sample
- Stops current interaction with device if one is underway
- Starts the measure process echoing if it is set on

SAMPLE LOCATOR (WS, DV, N, X, Y)

- Returns logical input value from MEASURE process without waiting for TRIGGER to fire

```
      SET LOCATOR MODE(WS, DV, SAMPLE, ECHO)
100   CONTINUE
      SAMPLE LOCATOR (WS, DV, N, X, Y)
      REDRAW BOAT AT (X, Y)
      GOTO 100
```

HOW DO WE STOP ?

```
      SET WINDOW (1, XWMIN, XWMAX, YWMIN, YWMAX)
      SET VIEWPORT(1, XVMIN, XVMAX, YVMIN, YVMAX)
      SET VIEWPORT INPUT PRIORITY (1, 0, HIGHER)
      SELECT NORMALIZATION TRANSFORMATION (1)
      SET LOCATOR MODE (WS, DV, SAMPLE, ECHO)
100   CONTINUE
      SAMPLE LOCATOR(WS, DV, N, X, Y)
      IF (N .EQ. 0) GOTO 200
      REDRAW BOAT AT (X, Y)
      GOTO 100
200   CONTINUE
```

MULTIPLE SAMPLE DEVICES

```
      SET VALUATOR MODE (WS, DV1, SAMPLE, NOECHO)
      SET VALUATOR MODE (WS, DV2, SAMPLE, NOECHO)

100   CONTINUE
      SAMPLE VALUATOR (WS, DV1, SIZE)
      SAMPLE VALUATOR (WS, DV2, ROT)

      EVALUATE TRANSFORMATION MATRIX
          (FX, FY, 0, 0, ROT, SIZE, SIZE, WC, MAT)

      SET SEGMENT TRANSFORMATION (BOAT, MAT)
      GOTO 100
```

EVENT MODE

SET LOCATOR MODE (WS, DV, EVENT, EC)

- Stops current interaction with specified device if one is underway
- Starts the MEASURE process echoing if it is set on
- Starts TRIGGER process. If already running, adds this device to TRIGGER's list of recipients
- EVENT REPORTS generated when TRIGGER fires and added to INPUT QUEUE

INPUT QUEUE

WORKSTATION IDENTIFIER	DEVICE NUMBER	LAST OF SIMULTANEOUS EVENTS	INPUT CLASS	LOGICAL INPUT VALUE

Temporal Order

Functions are provided to interrogate queue and manipulate it

AWAIT EVENT

CALL GWAIT (TOUT, WS, DVCLASS, DV)

- If the input queue is not empty, remove first event report and copy logical input value into current event report.

  Return WS = workstation identifier
  DVCLASS = device class
  DV = device number

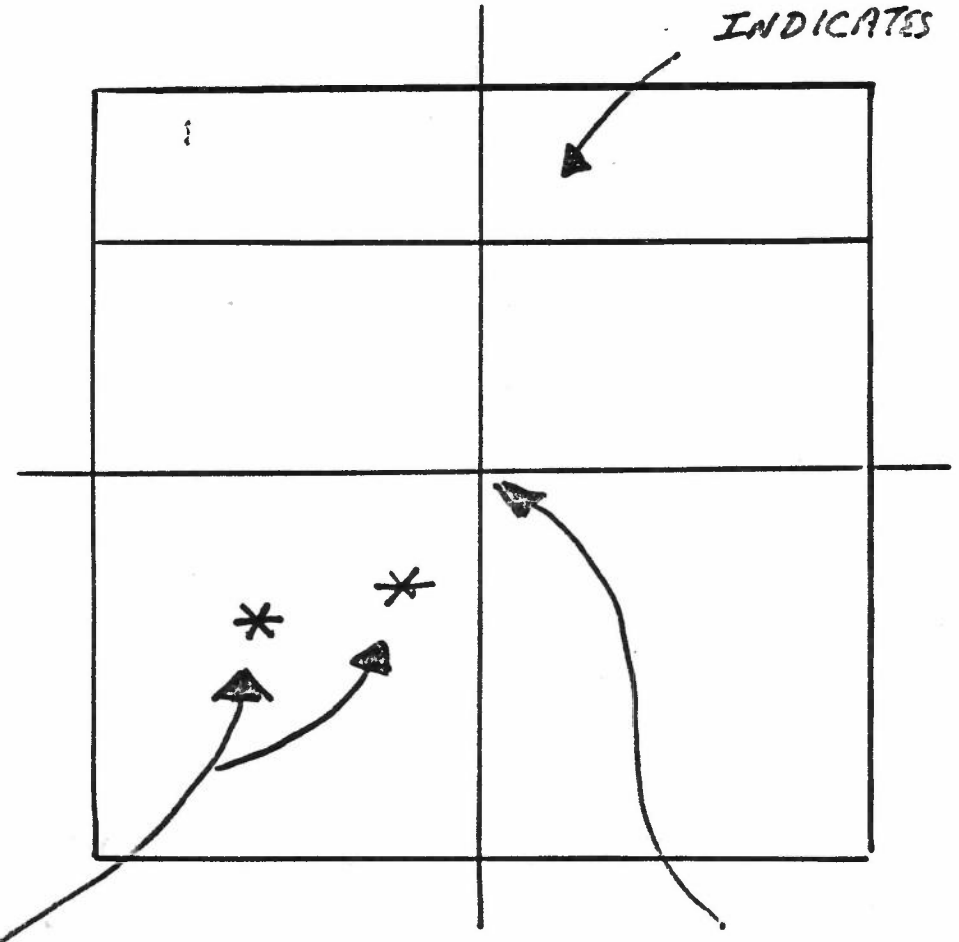
- If input queue is empty, wait until either event occurs or TOUT seconds pass. In first case, action as above.
  If no event occurs, DVCLASS is returned as NONE

GET

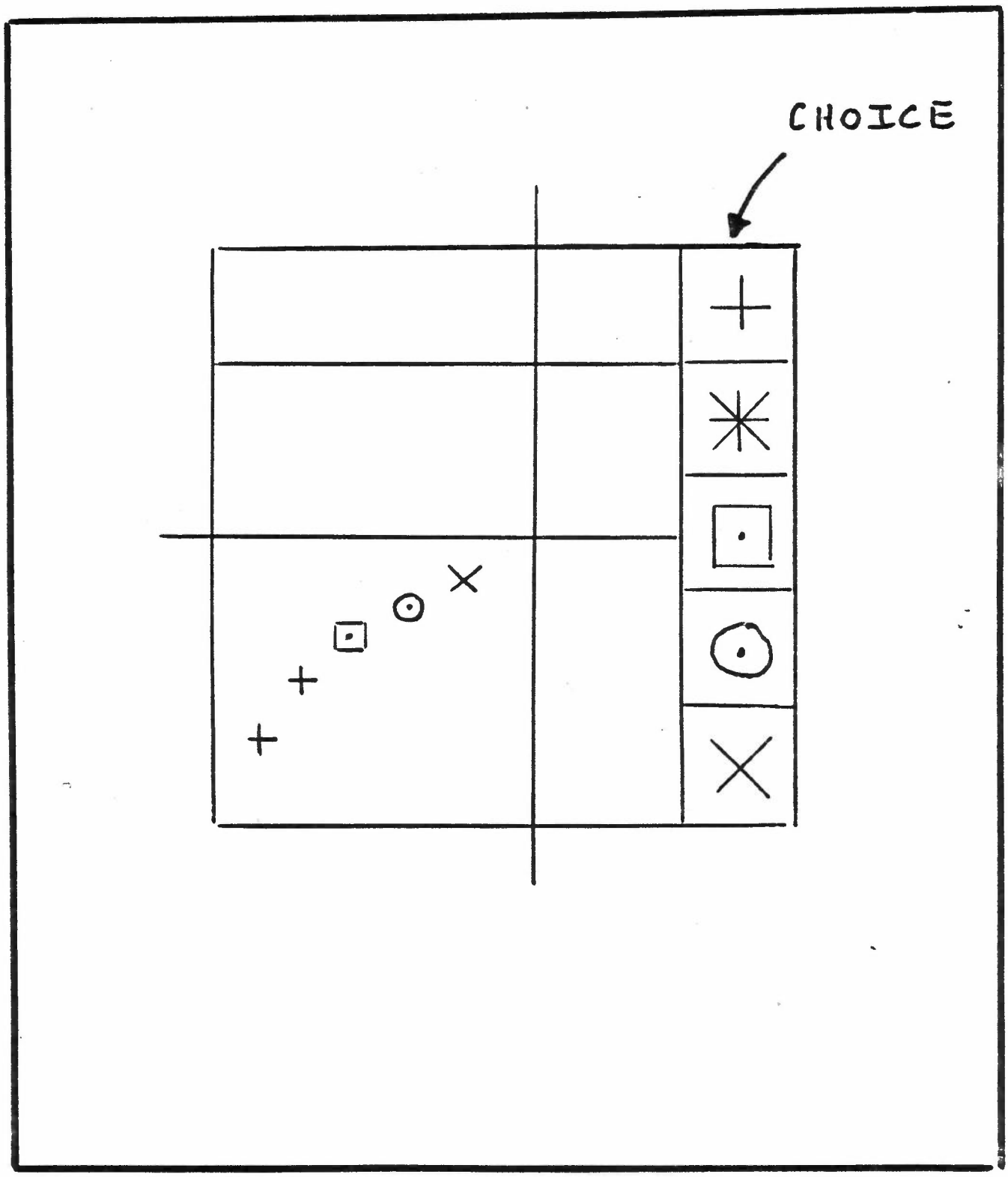
Appropriate GET function is used to read logical input value from current event report

```
GET LOCATOR (N, X, Y)
  CALL GGTLC (N, X, Y)

GET VALUATOR (VAL)
  CALL GGTVL (VAL)

GET CHOICE (STAT, CHNR)
  CALL GGTCH (STAT, VAL)
  Status = STAT (OK or NOCHOICE)

GET PICK (STAT, SEG, PICKID)
  CALL GGTPK (STAT, SEG, PICKID)
  Status = STAT (OK or NOPICK)

etc
```

TO INPUT A POLYLINE



```
      SET WINDOW(1, XMIN,XMAX,YMIN, YMAX)

      SET VIEWPORT(1, 0, 1, 0, 0·8)
      SET WINDOW(2,X2MIN,X2MAX, Y2MIN, Y2MAX)
      SET VIEWPORT(2,0, 1, 0·8, 1)
      SET VIEWPORT INPUT PRIORITY (0,2,LOWER)
      SELECT NORMALIZATION TRANSFORMATION (1)
      SET LOCATOR MODE (WS, DV, EVENT, ECHO)
      NPTS =0

100   CONTINUE
      AWAIT EVENT(6000, WST, CLASS, DEV)
      GET LOCATOR (NT, X(1), Y(1))
      IF (NT .EQ. 2) GoTo 200
      NPTS = NPTS + 1
      XLIST(NPTS) = X(1)
      YLIST (NPTS) = Y(1)
      POLYMARKER ( 1, X, Y )
      GoTo 100
200   IF (NPTS .GE. 2) POLYLINE(NPTS, XLIST, YLIST)
```


CHOICE

```
      SET POLYMARKER INDEX (1)

      SET LOCATOR MODE (WS, DV1, EVENT, ECHO)
      SET CHOICE MODE (WS, DV2, EVENT, ECHO)

100   CONTINUE
      AWAIT EVENT (600, WST, CLASS, DEV)
      IF ( CLASS .EQ. NONE ) STOP
      IF (CLASS .EQ. LOCATOR) GOTO 200
      GET CHOICE (STAT, CH)
      SET POLYMARKER INDEX (CH)
      GOTO 100

200   GET LOCATOR (NT, X(1), Y(1))
      POLYMARKER (1, X, Y)
      GOTO 100
```

FLUSH DEVICE EVENTS

CALL GFLUSH (WS, DVCLASS, DV)

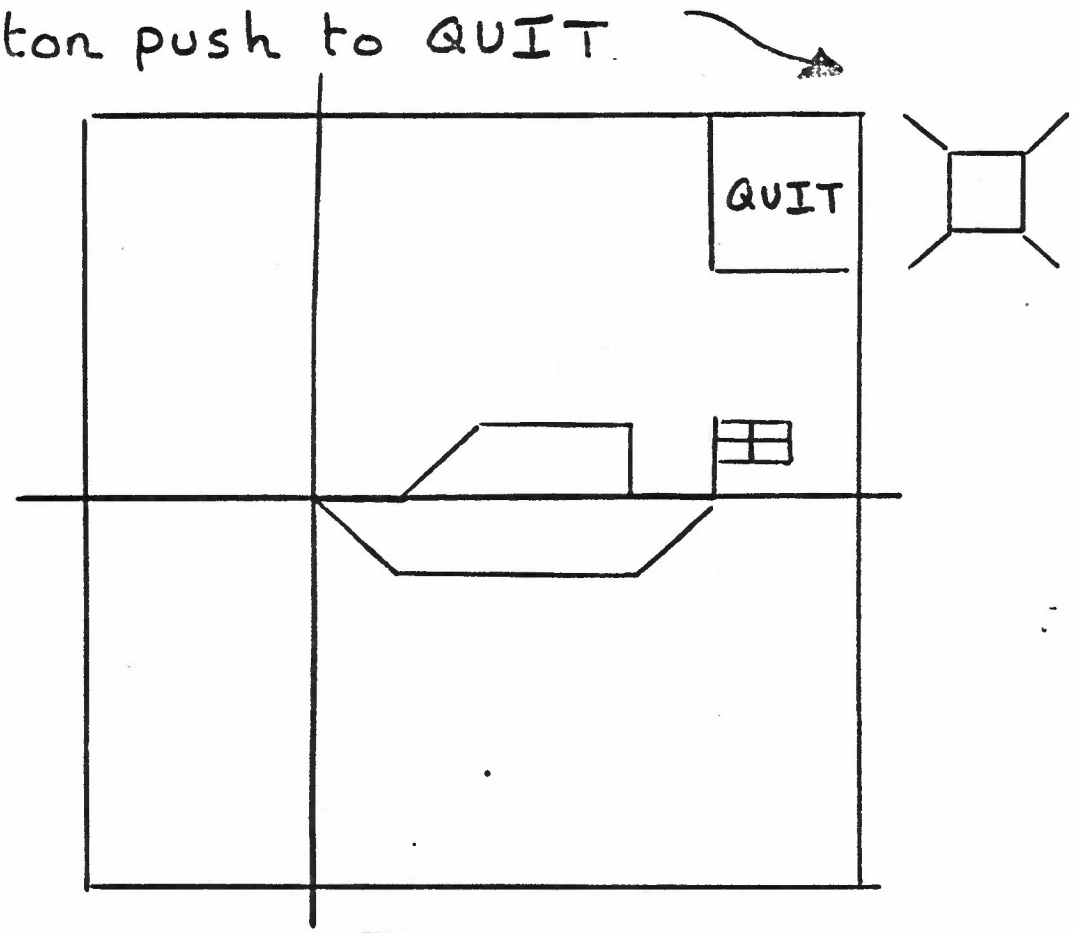
Removes all entries in input queue from specified device

```
      SET LOCATOR MODE (WS, DV, EVENT, ECHO)
      NPTS = 0
100   CONTINUE
      AWAIT EVENT (6000, WST, CLASS, DEV)
      IF (NT .EQ. 2) GOTO 200
      NPTS = NPTS + 1
      XLIST (NPTS) = X(1)
      YLIST (NPTS) = Y(1)
      POLYMARKER (1, X, Y)
      IF (NPTS .EQ. MAX) GOTO 300
      GOTO 100
200   IF (NPTS .GE. 2) POLYLINE (NPTS, XLIST, YLIST)
      - - - - - - -
300   POLYLINE (MAX, XLIST, YLIST)
      FLUSH DEVICE EVENTS (WS, LOCATOR, DV)
```

MIXED MODES

CHOICE DEVICE

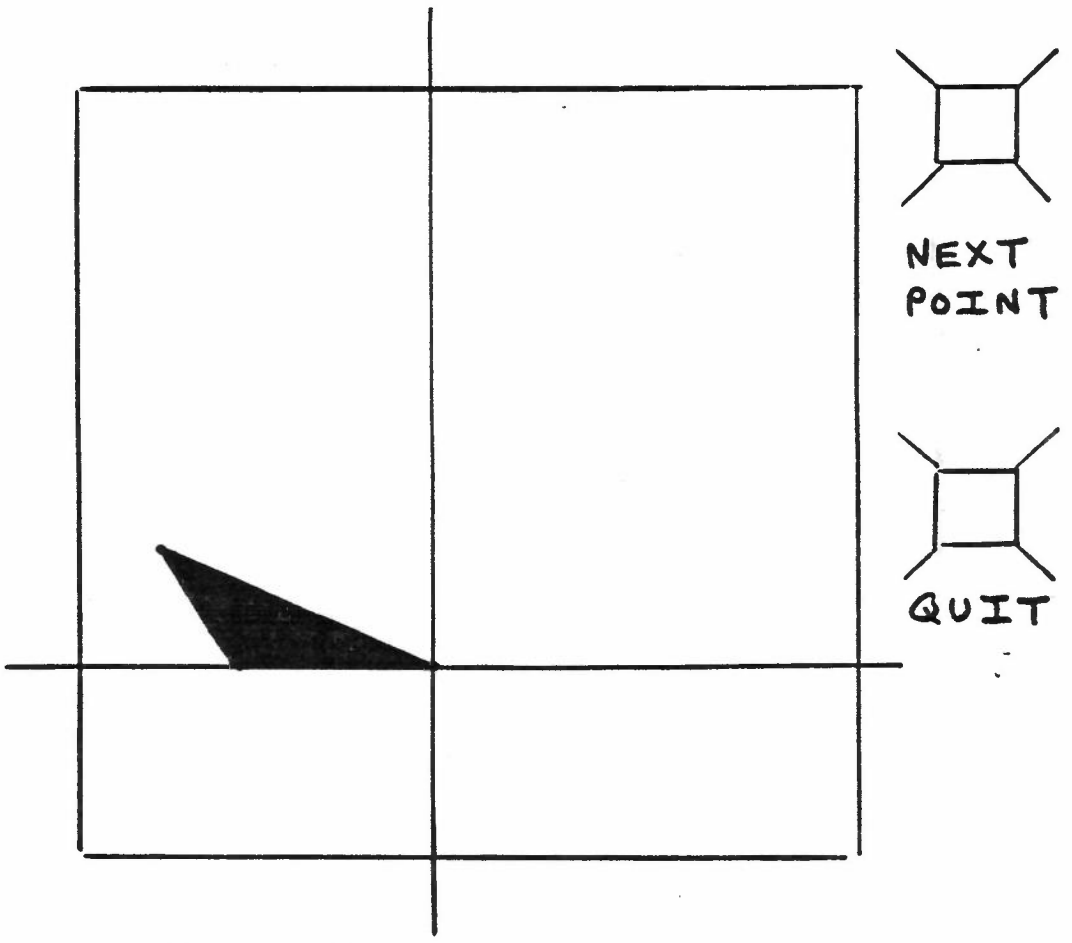
Button push to QUIT



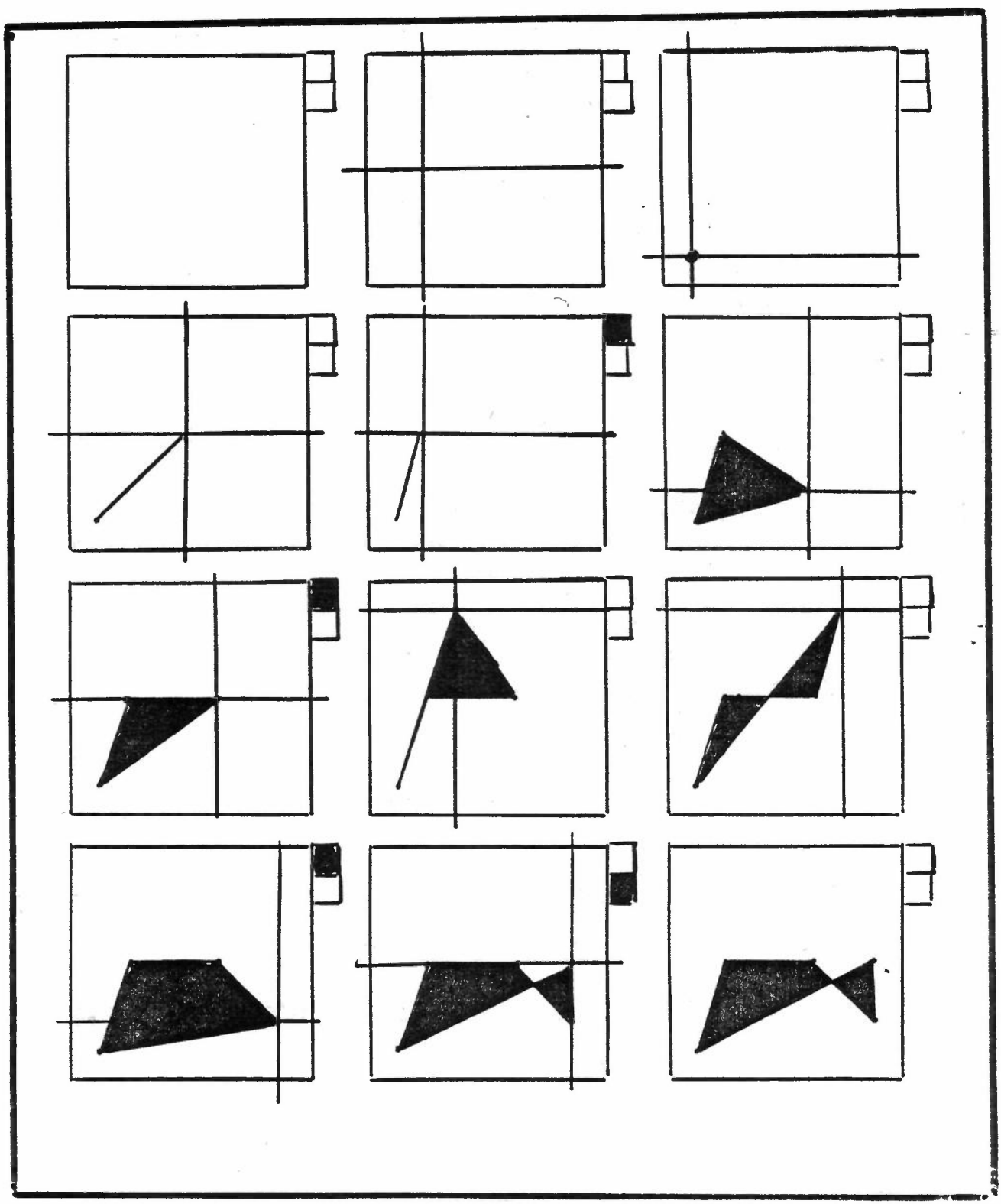
- BOAT follows LOCATOR as it is sampled
- Final position decided by button hit

```
      SET LOCATOR MODE (WS,DV1,SAMPLE,ECHO)
      SET CHOICE MODE (WS,DV2,EVENT,NOECHO)

100   CONTINUE
      SAMPLE LOCATOR (WS,DV1,NT,X,Y)

      REDRAW BOAT AT (X,Y)

      AWAIT EVENT (0,WST,CLASS,DEV)

      IF (CLASS .NE. CHOICE) GOTO 100

      FLUSH DEVICE EVENTS (WS,CHOICE,DV2)
```

AREA DEFINITION BY LOCATOR



- Set of points already input define an area
- LOCATOR sampled for next point. New area shown.



```
      SET LOCATOR MODE (WS, DV1, REQUEST, ECHO)
      REQUEST LOCATOR ( WS , DV1 , ST, NT, XAR(1), YAR(1) )
      NP = 1

      SET LOCATOR MODE (WS, DV1, SAMPLE, ECHO)
      SET CHOICE MODE (WS, DV2, EVENT, NOECHO)

      CREATE SEGMENT (AREA)
      CLOSE SEGMENT

100   CONTINUE
      NP = NP + 1
      XAR (NP+1) = XAR (1)          closes polygon
      YAR (NP+1) = YAR (1)

200   CONTINUE
      SAMPLE LOCATOR (WS, DV1, NT, XAR(NP), YAR(NP) )

      DELETE SEGMENT (AREA)
      CREATE SEGMENT (AREA)
      FILL AREA ( NP+1, XAR, YAR )
      CLOSE SEGMENT

      AWAIT EVENT ( 0, WST, CLASS , DEVICE )

      IF (CLASS .NE. CHOICE) GOTO 200

      GET CHOICE ( STAT , CH )
      IF ( STAT .EQ. NOCHOICE ) GOTO 200

      IF (CH .EQ. 1) GOTO 100

      FLUSH DEVICE EVENTS (WS, CHOICE, DV2)

      SET LOCATOR MODE (WS, DV1 , REQUEST , ECHO)
```

INTERACTION IN GKS

- While interaction in GKS underway, operator controls the measure
- When no interaction underway, measure does not exist

REQUEST

Single call to REQUEST contains entire interaction

SAMPLE and EVENT

Interaction starts with:

SET ___ MODE

Continues until SET___MODE starts another interaction or

SET ___ MODE (, , REQUEST,)

stops current interaction

INITIALISE LOCATOR

```
CALL GINLC (WS, DV, N, X, Y, PE,
    XMN, XMX, YMN, YMX, IMX, IA)
```

Initialises the locator device specified by WS and DV

- LOCATOR measure initialised to (X, Y) relative to normalization transformation N
- PE specifies prompt/echo type
- XMN, XMX, YMN, YMX gives echo area
- IA(1) → IA(IMX) is input data record for additional information
- Defaults provided if INITIALISE LOCATOR not called

GOOD PRACTICE

When inputting a set of locations, initialise locator at last position input

```
100 CONTINUE
    REQUEST LOCATOR (WS,DV,ST, N, X, Y)
    REDRAW BOAT AT (X,Y)

    INITIALISE LOCATOR (WS,DV,N,X,Y,
     PE, XMN,XMX,YMN,YMX, IMX,IA)

    GOTO 100
```

NOTE initial value by the program must be legal - within the relevant window

INITIALISE VALUATOR

```
CALL GINVL(WS,DV,VAL,PE,XMN,XMX,
           YMN,YMX,LOVAL,HIVAL,IMX,IA)
```

INITIALISE CHOICE

```
CALL GINCH(WS,DV,ISTAT,ICH,PE,
     XMN,XMX,YMN,YMX,IMX,IA)
```

INITIALISE PICK

```
CALL GINPK(WS,DV,ISTAT,SEG,PICKID,
     PE,XMN,XMX,YMN,YMX,IMX,IA)
```

INITIALISE STRING

```
CALL GINST(WS,DV,LSTR,STR,PE,XMN,XMX,
     YMN,YMX,BUFLEN,INCURPOS,IMX,IA)
```

INITIALISE STROKE

```
CALL GINSK(WS,DV,N,X,Y,PE,XMN,XMX,
     YMN,YMX,BUFLEN,IMX,IA)
```

PROMPT AND ECHO TYPE		
LOCATOR	2	crosshairs
	3	tracking cross
	4	rubber band lines
	5	rubber band rectangles
	6	digital representation
PICK	2	highlight pick group
	3	highlight segment
VALUATOR	2	dial
	3	digits
CHOICE	2	lamps
	3	menu of strings
	4	string commands
	5	symbol menu
STRING	1	string in echo area
STROKE	2	digits
	3	markers
	4	lines
ALL BUT STRING	1	implementation dependent

PROMPT AND ECHO TYPE

Values less than zero are implementation dependent

Zero is invalid.

Other positive values are subject to registration

ECHO AREA

Two main types of prompt/echo:

- Can be anywhere or not on display at all

LOCATOR	crosshairs
PICK	highlighting
CHOICE	lamps
STROKE	markers

- Displayed at a particular place

LOCATOR	digital representation
VALUATOR	digits
CHOICE	menu
STRING	
STROKE	digits

INITIALISE echo area parameters specify this place

ECHO AREA specified in device coords

INPUT DATA RECORD

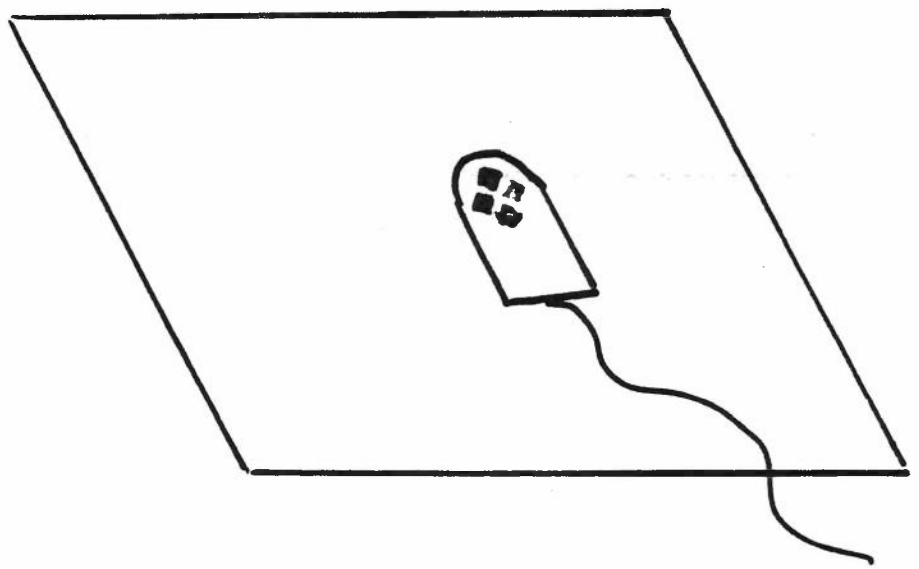
Some logical input devices allow further control over the interaction

- SIZE OF TRACKING CROSS
- CLOSENESS WHEN TABLET USED TO PICK
- COLOUR FOR PICK HIGHLIGHTING
- RUBBER BAND LINETYPE

Additional information of this type is stored here in the data record

SIMULTANEOUS EVENTS

Triggers may be defined such that several events are entered in input queue from a single trigger firing



- BUTTON HIT causes
  - (1) LOCATOR EVENT
  - (2) CHOICE EVENT

INQUIRE MORE SIMULTANEOUS EVENTS

CALL GQSIM (ERRIND , ANYMORE)

Input queue is checked to see if any events exist triggered by same firing as last one read

ANYMORE = MORE or NOMORE

INQUIRE INPUT QUEUE OVERFLOW

CALL GQIQOV (ERRIND, EWS, ECLASS, EDV)

Checks if input queue overflowed

ERRIND = 0 , queue has overflowed

No more events added to queue until it is emptied

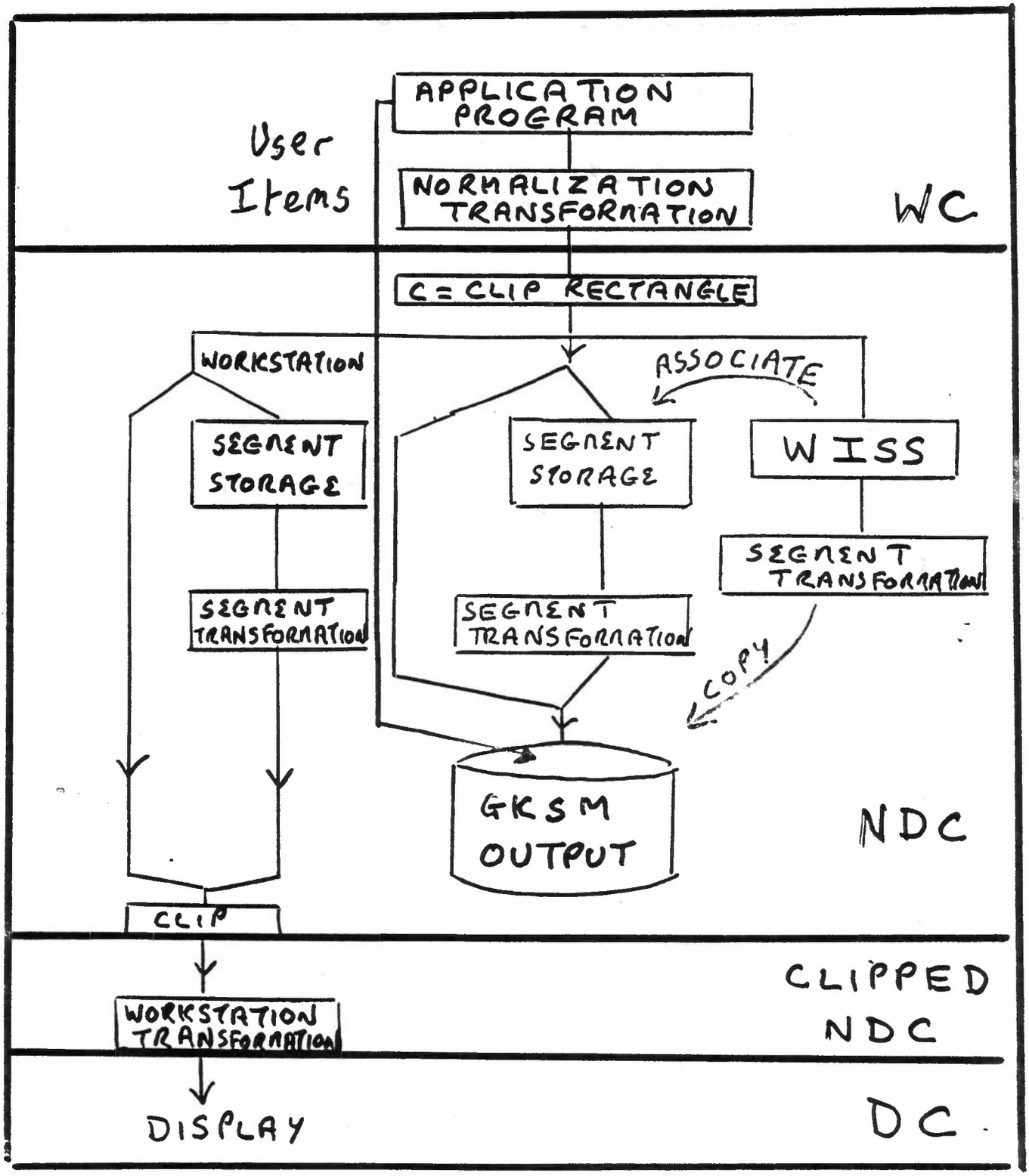
LEVELS

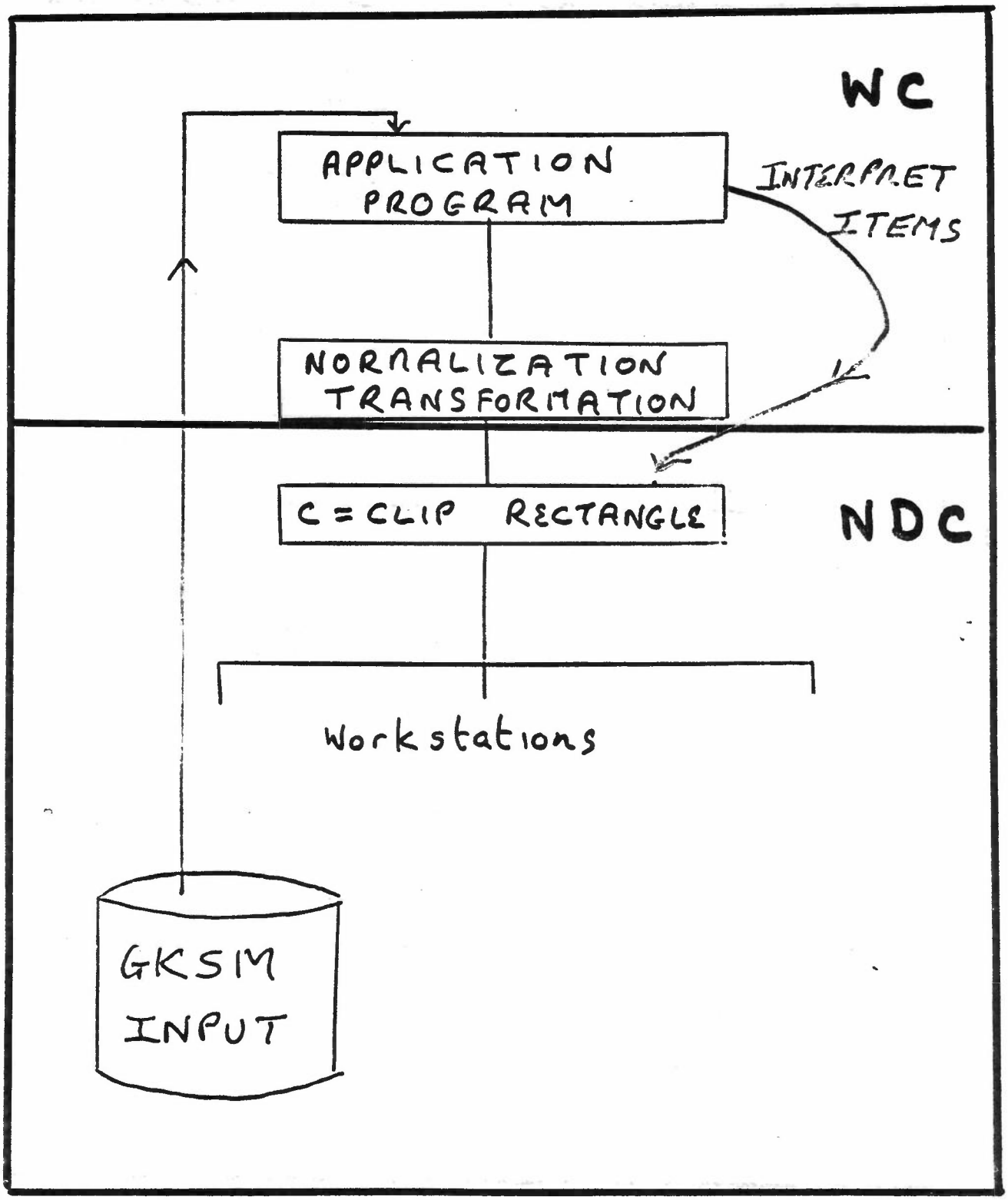
Output Level \ Input Level	a	b	c
m (ANSI)	Only individually set attributes	m^a with o^b style input	m^a with o^c style input
0	No graphic input Minimal Control Predefined bundles only	REQUEST input No PICK input	SAMPLE and EVENT input No PICK
1	Full Output Full bundle setting Segments	o^b plus REQUEST PICK	o^c plus all PICK input
2	Workstation Independent Segment Storage	2^a and 1^b	2^b and 1^c

GKS METAFILE

GKS Metafile can be used for:

- transporting graphical information between systems
- transporting graphical information from place to place (e.g. magnetic tape)
- transporting graphical information from one GKS application to another
- storing accompanying non-graphical information


APPLICATION PROGRAM
User Items
NORMALIZATION TRANSFORMATION
WC
C = CLIP RECTANGLE
WORKSTATION
ASSOCIATE
SEGMENT STORAGE
SEGMENT STORAGE
WISS
SEGMENT TRANSFORMATION
SEGMENT TRANSFORMATION
SEGMENT TRANSFORMATION
COPY
GKSM OUTPUT
NDC
CLIP
CLIPPED NDC
WORKSTATION TRANSFORMATION
DISPLAY
DC


WC
APPLICATION PROGRAM
INTERPRET ITEMS
NORMALIZATION TRANSFORMATION
C = CLIP RECTANGLE
NDC
Workstations
GKSM INPUT

WRITING METAFILES

- Metafiles treated as special workstation
- User function WRITE ITEM TO GKSM allows application program to write

READING METAFILES

- Treated as a workstation
- GET ITEM TYPE FROM GKSM
  delivers item type and item data record of current item
- READ ITEM FROM GKSM
  copies current item into data area and reads next item
- INTERPRET ITEM
  interprets supplied item data record

GKS IMPLEMENTATIONS

	COMPANY	LEVEL	LANGUAGE BINDING	COMMENTS
1	RAL/ICL	2b	FORTRAN	Portable version ported many systems
2	GTS-GRAL	2c	FORTRAN	Fast implementation Good characteristics
3	NOVA GRAPHICS	2b	FORTRAN PASCAL C	Distributed Implementation
4	Amsterdam CWI	2c	FORTRAN C	Widely available. Sold by others
5	TEKTRONIX	2b	FORTRAN	Non-standard input for Tektronix devices
6	CEEGEN	2b	FORTRAN	Modelling system on top
7	WHITECHAPEL	2b	FORTRAN PASCAL C	Based on QMC
8	PRIOR DATA SCIENCES	2b	C	Non-standard segment functions
9	PRECISION VISUALS	2b	FORTRAN	Rich implementation
10	DATAPLOTTING SERVICES	2b	FORTRAN PASCAL C	Interfaces to CALCOMP & PLOT10
11	RAMTEK	2b	FORTRAN	Hardware assist
12	VISUALENGINEERING		FORTRAN	Available on Pyramid
13	ADVANCED TECHNOLOGY CTR	2c	FORTRAN C	
14	UNIRAS	2b	FORTRAN	Many application packages on top
15	DEC	0b	FORTRAN	

	COMPANY	Level	LANGUAGE BINDING	COMMENTS
16	IBM/Graphic Software Systems		FORTRAN & BASIC	On IBM PC
17	INFOLYTICA	m b	FORTRAN	CAD Systems being built on top
18	AED - GKS	2c	FORTRAN	
19	CMC - BOMBAY	2b	FORTRAN	Also have GKS-3D
20	SYSGRAPH	2b	FORTRAN	
21	SYSTEM SIMULATION	1a	FORTRAN	
22	XGKS	2b	FORTRAN	Good Hungarian implementation
23	HARRIS		ADA	
24	QUEEN MARY COLLEGE	2b	FORTRAN	
25	DATA GENERAL	2b	FORTRAN	

IMPLEMENTATION PHILOSOPHIES

- POSITIONING THE DI/DD INTERFACE
  - Thin back end
    - device drivers simple
    - intelligent devices unable to use features
  - Thin front-end
    - simple devices have complex drivers
    - intelligent devices supported
  - Negotiation
    - a la GINO-F
    - back-end can refuse most requests
  - Device Dependent Tool Set
    - thin front end with tool set for device drivers
  - Layered
    - several internal interfaces
- IMMEDIATE/DELAYED EXECUTION
  - Delay attribute changes etc until necessary
- SEGMENT STORAGE
  - Central Segment Store (WDSS with WISS)
  - Distributed Segment Store (WISS in workstation)

GKS IMPLEMENTATIONS
GLOBAL DIFFERENCES

- GKS Level
- Number of Normalization Transformations
- Available workstation types
- Max number workstations open, active, associated with a segment
- Size of input queue
- Number of fonts, GDPs, ESCAPES
- ASF initial settings
- Font definitions

WORKSTATION DIFFERENCES

- Device coordinates
- Available linetypes etc
- Number of GDPs
- Colour Table size
- Input devices
- Prompt/echo types
- Input buffer sizes (STRING/STROKE)
- etc
-

MINIMUM SUPPORT REQUIRED

	0a	2c
Linetypes	4	4
Predefined bundles polyline	5	5
Settable bundles	–	20
Character heights	1	1
Stroke fonts	0	2
* Predefined patterns	1	1
Settable patterns	–	10
* Hatch Styles	3	3
Settable Norm Transf	1	10
* Segment Priorities	–	2
Input queue length	–	20
Max STRING buffer	–	~~32~~
MAX STROKE buffer	–	64

* CAN DECLINE TO DO