AD-A279 734

**S** DTIC
ELECTE
JAN 03 1994
**A**

# FACULTY OF COMPUTER SCIENCE

94-14281
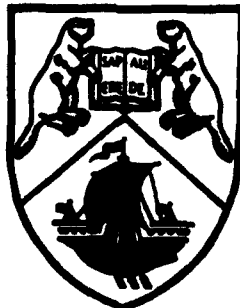
UNIVERSITY OF NEW BRUNSWICK

FREDERICTON, N.B. CANADA    E3B 5A3

93 12 29015

*ΦSCUBA*

A BUFFERED CORE GRAPHICS SYSTEM

BY

Uday G. Gujar

School of Computer Science
University of New Brunswick
Fredericton, N.B.


Aragam R. Nagesh

School of Computer Science
University of New Brunswick
Fredericton, N.B.

TR79-016, June 1979

## ABSTRACT

The graphics community is recently taking a close look at graphics standardization prompted by issues such as portability of software, application program structure, diverse hardware, etc. Through these efforts, the Graphic Standards Planning Committee of ACM/SIGGRAPH has proposed a standard. This standard advocates four levels of the Core system, namely basic, buffered, interactive and complete. This paper describes a Level 2 (i.e. buffered) implementation of the Core system in APL, referred to as $\phi SCUBA$ (APL BUffered Core System). The details of the structure and organization of $\phi SCUBA$ are given. The implementation is highly modular in nature, provides both two and three dimensional capabilities with several types of projective transformations and supports full segmentation capabilities. Several examples illustrating the use of the system are included. The interactive nature of APL is found to be attractive. Some deviations from the Core system have been incorporated. These include a modular hardcopy interface to produce graphics on plotters etc. and a facility to retain world coordinates of the objects. The system, though appearing to be satisfactory, has to undergo further testing to gain user confidence.

## 1. INTRODUCTION:

ΦSCUBA (APL BUffered Core System) is an APL based system for generating and maintaining graphics displays. The system is implemented under APLSV to run on an IBM 370/3032 at the University of New Brunswick. It has been designed and developed to meet the functional capabilities of the Level 2 Core System proposed by the Graphics Standards Planning Committee of ACM/SIGGRAPH in July of 1977 [GSPC77].

The main objectives in developing ΦSCUBA are device independence, adherence to CORE specifications and maintenance of pictures as simple user files. Device independence implies that it should be possible to draw on any graphics devices currently supported, which include storage tubes, a drum plotter, an electrostatic plotter and line printers. Close adherence to CORE specifications is the second design goal; Level 2 is chosen because of hardware limitations. Thirdly, maintenance of pictures as user files is prompted by the fact that the application programmer is essentially an APL user and hence has to manage the available workspace area. In terms of capabilities, the system is capable of producing 2D or 3D pictures or 2D plots using graphics primitives like MOVE and DRAW for the creation of pictures. Also, full general viewing specifications are available to create several images (projections) of the same object. Finally, facilities to interact with the objects and/or images in terms of changes or transfers to other devices are provided.

In the following sections a brief discussion of the structure, implementation and the highlights of ΦSCUBA is presented.

## II. STRUCTURE OF $\phi SCUBA$:

The complete system has been organized in terms of groups of APL functions. Three essential groups (CORE1, CORE2, CORE3) form the bulk of Level 2 Core System specifications and CORE4 forms the modelling system for $\phi SCUBA$. Additional groups comprise the global variables, device drivers and device dependent routines, and the backbone of the TSIO filing facility. Finally, a hardcopy interface forms a separate group which helps in transferring pictures to plotters and printers not directly supported under the APLSV system.

Figure 1 depicts the general organization of $\phi SCUBA$ and the overall flow of information in the system. The groups have been combined to form modules for the sake of clarity.

In conforming to the CORE specification [GSPC77], the modelling system is separated from the viewing system. This dichotomy not only helps in achieving a clean system but also saves valuable workspace area in APL since the user can dynamically release storage used by the modelling system functions.

Module 'C' comprising of CORE1, CORE2 and CORE3 forms the complete device independent Level 2 Core System. The functions in CORE1 implement all the output primitives, viewing transformations, and the general 3D clipping. CORE2 forms the segment operations submodule and has functions to generate and maintain picture segments and associated data structures. A slight deviation from CORE specifications is to be found here in terms of object data structures and retaining of objects in
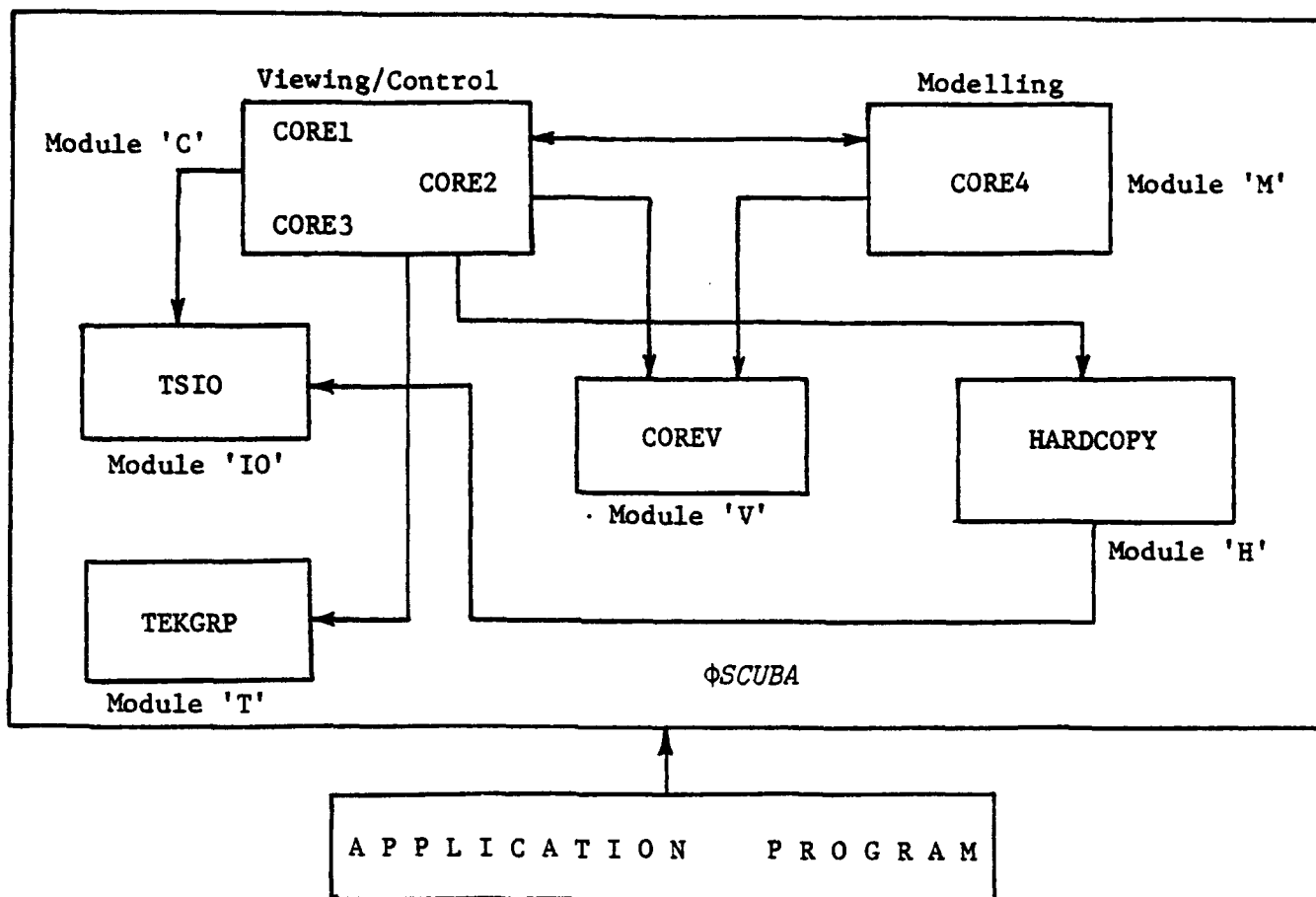
Figure 1: Organization of $\phi SCUBA$

addition to the images. The details are discussed later under imple-
mentation. Finally, CORE3 implements the control functions of the CORE
specifications such as setting and inquiry of attributes, selecting
viewing control parameters, etc. Instead of leaving the application
programmer to make individual function calls (as specified by GSPC) for
various settings or inquiries, one interactive function for each set of
attributes (say, viewing or control or primitive) is written for this
purpose. This approach is found to be of valuable use in the imple-
mentation of some of the CORE functions.

Module 'M' is the modelling functions module used to describe the orientation or position of the object in the world coordinate system. This module is used in conjunction with the module 'C' since the initial object definition is to be through primitive invocations in an open segment.

Module 'V' is entirely made up of global variables used by the system. It contains such items as representative data structures for objects and images, list of viewing parameters, some 'HELP' documentation, etc.

Module 'IO' consists of essential TSIO routines for creation and maintenance of retained segments. Any errors that are detected in this module are passed on to the higher level routines in module 'C' for reporting.

The module 'T' consists of device drivers (currently only TEKTRONIX 4015 storage display drivers) and thus form the device dependent portion of the ΦSCUBA system. The functions of this module can be used for line drawing, character writing, screen erasing, etc. It also includes some global variables like character stroke table, screen dimensions and control characters.

The hardcopy interface in module 'H' consists of a specialized set of routines which transfer pictures drawn on the TEKTRONIX display to any of the other plotting devices not supported under APLSV. The technique used is to record all the 'pen movements' in a TSIO file as the picture is constructed and use this file as an input to the 2D

plotting package [GUJA72, GUJA76] available under the general operating system. The details of this module are explained under implementation.

The overall structure of $\phi SCUBA$ lends itself to easy application programming. A majority of the functions are niladic and any input expected of the user is handled interactively via a conversation. Thus the application programmer's knowledge of APL has essentially been kept to the minimum. Error reporting, although done through individual functions, is organized so as to avoid occurrences of suspended functions and holding of storage space. A knowledgeable APL programmer, however, can make use of the facilities of $\phi SCUBA$ much more efficiently by defining his/her own functions using the $\phi SCUBA$ functions. By doing so, the user can obtain the intended results in a faster way.

## III.  IMPLEMENTATION DETAILS:

The implementation langauge being APL, its dynamic array handling capabilities are extensively used. All the data are represented as real arrays thus maintaining a consistent storage structure. For retaining purposes the available TSIO facilities [UNB74] under APLSV are used. The character strings appearing in text are converted to equivalent real constant by using the encoding operator of APL. This proves to be simple and economical on storage. Figure 2 summarizes the structure of the various arrays used for storing objects and their images; the following notation is used in that Figure:

XW,YW and ZW are the world coordinate dimensions of a point,

XI,YI are the transformed dimensions representing the image of a

point,

| 0 OR 1 | XW | YW | ZW |
|--------|----|----|----|
|        |    |    |    |
| 0 OR 1 | XW | YW | ZW |

N × 4   0 for MOVE
        1 for DRAW

a)  Wire frame drawing in world coordinates.

| SYMBOL# | XW | YW | ZW |
|---------|----|----|----|
|         |    |    |    |
| SYMBOL# | XW | YW | ZW |

N × 4   SYMBOL# = Index in the
        APL symbol
        set

b)  Markers in world coordinates.

| XW | YW | ZW | HTW | WDW | XSP | YSP |
|----|----|----|-----|-----|-----|-----|
| XI | YI | R | XSPI | YSPI | HTI | WDI |
| LENGTH | TEXT CODE | TEXT CODE | TEXT CODE | TEXT CODE | TEXT CODE | |

c) Text Vector - World definition and/or corresponding image.

EP1                    EP2

| XI | YI | XI | YI |
|----|----|----|----|
|    |    |    |    |
| XI | YI | XI | YI |

N × 4   EP1 - End Point 1
        EP2 - End Point 2

d)  Image of wire frame drawing.

| SYMBOL# | XI | YI |
|---------|----|----|
|         |    |    |
| SYMBCL# | XI | YI |

N × 3   SYMBOL# = Index in the APL
        symbol set

e)  Image of markers.

Figure 2:  Data structures for objects and images

XSP,YSP,XSPI,YSPI are the spacing parameters in a line of text in

the world and image coordinates respectively,

HTW,WDW,HTI,WDI are the character size parameters in world and

image coordinates respectively and

R is the angle, in radians, of inclination of the text string; it

is calculated internally.

At Level 2, the Core System provides everything except detectability
of segments, input primitives and image transformations. Thus, all the
output primitives and their attributes (excepting some device dependent
ones like text font, color and highlighting) are implemented. In imple-
menting the text primitives, only the low quality text is chosen. The
high quality text is too expensive to implement and the medium quality
text, it is felt, does not offer any great advantage over the low qual-
ity. In fact, the medium quality text has the disadvantage of overlapped
characters (see examples in [GSPC77]). A software stroke chracter gener-
ator to produce standard APL character set of 120 symbols is provided.

To implement the viewing transformations, the synthetic camera
analogy of the Core System [NEWM78, GSPC77] is used. The user could
choose a particular view from the six possible views: perspective,
oblique, isometric, top, front and side. The latter four are particu-
lar cases of orthographic projections; certain viewing parameters are
set automatically by the system for these views. The user must be
knowledgeable of the viewing transformations to set the particular para-
meters for getting two or three point perspectives or a particular cab-
inet or cavalier projection. The mathematics used in this implementation

is based on the paper by Carlbom and Paciorek [CARL78] and discussions
by Rogers and Adams [ROGE76] and Newman and Sproull [NEWM73].

A general 3D clipping is employed for world coordinate clipping
where the particular viewing volume (infinite or truncated pyramid or
parallelopiped) is derived from the viewing parameters. An algorithm
given by R.F. Puk [PUK77] is implemented for achieving the above clip-
ping. The clipping (both window and depth) is user controllable and the
options have been included as control parameters along with the type of
world coordinate system (left or right).

Picture segmentation and naming facility of the Core System is im-
plemented through the segment operations submodule and the TSIO module.
A segment is created by invoking the CREATE function. The created seg-
ment can be either named or unnamed and can be retained or nonretained.
Only nonretained segments can be nameless. Any created segment results
in the creation of an entry in a system maintained segment directory
(see Figure 3). An entry in the segment directory shows an eight
character segment name (padded with blanks to the right, if needed) and
three flags associated with visibility, type and residence. The visi-
bility and the type are directly the segment attributes of the Core
System. The residence flag is a special flag which indicates the pre-
sence or absence of the world coordinate definition of a particular
segment in the APL workspace. This additional feature is prompted by
the facility to retain objects in addition to the images in a segment.
More details of this new facility are discussed in the following section.
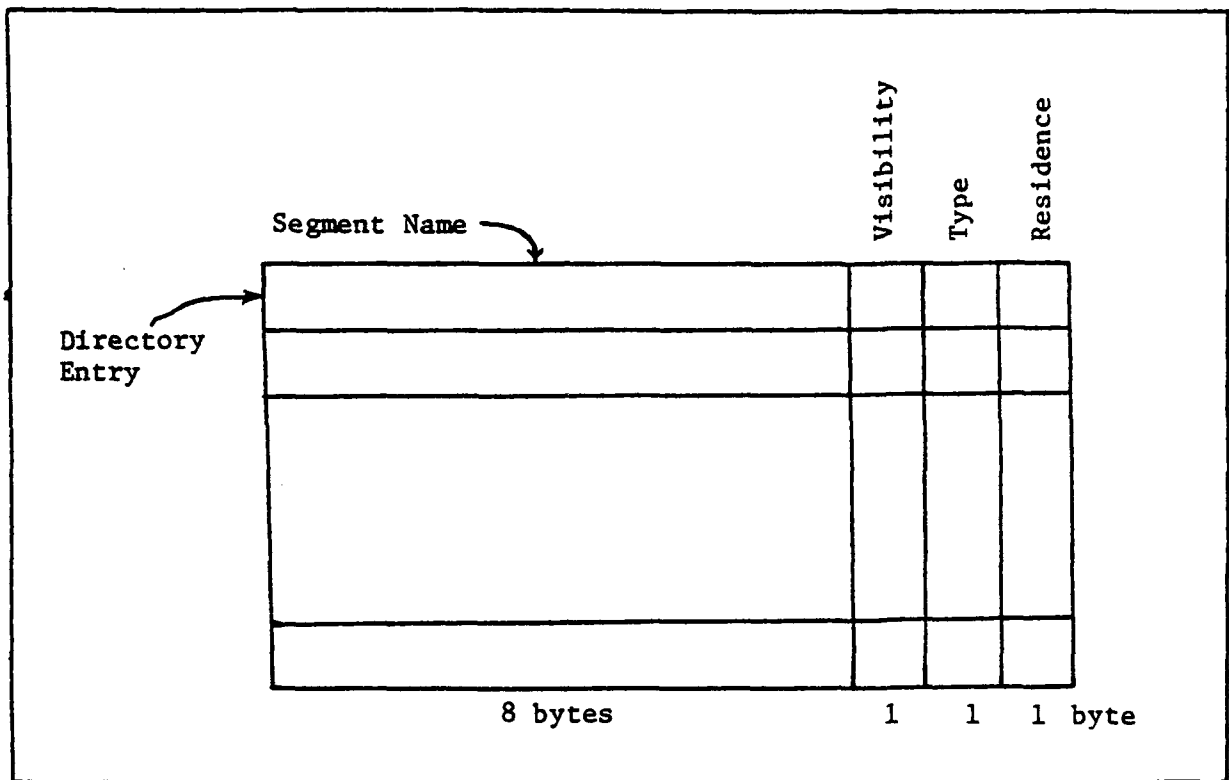
Figure 3:  Segment Directory

For each created segment, there can possibly be two TSIO files - one
for image and one for object.  In addition, each user will have his/her
segment directory as a TSIO file which is augmented each time a segment
operation affecting the segment directory is performed.  At any time
the user can inquire the contents of his/her segment directory by means
of a LISTDIRY function.  Independent loading and storing of the dir-
ectory is also possible by means of LOADDIRY and STORDIRY functions.

The double buffering required in Level 2 of the Core System [GSPC77]
has been achieved through RENAME function.  In addition, a NAME function
has been provided which can be used to name an unnamed segment.  This
feature helps the user to work on an unnamed segment initially until
the segment appears to be satisfactory; at this time a name can be

attached to retain the segment, if desired.

As mentioned before, the CORE functions are implemented in an inter-
active mode via a conversation with the user. Typically, setting up of
the viewing parameters, segment attributes and primitive attributes is
handled via niladic functions such as SETVIEW or SETPRAT. Similarly,
inquiry of attribute values is handled via the INQUIRE function. Clearly,
this scheme can be employed either separately or inside other functions
developed by an application programmer.

Error handling has been achieved through individual functions them-
selves rather than through a separate error handler. However, depending
on the severity of an error, the user is either prompted to continue
with a corrective action (e.g. duplicate segment name error) or to
terminate current activity through a return to the highest level of
function call. The latter is achieved in APL very easily through exe-
cution of a niladic branch arrow. Care has been taken to see that the
user is not confronted with incomprehensible error situations. Even
in the worst case of a workspace overflow, an elementary knowledge of
APL and of the names of the modules of the system will help the user to
dynamically expunge some variables to make room in the workspace in
order to continue execution.

The implementation ˆf an auxiliary hardcopy interface for the $\phi SCUBA$
system is necessitated by the nonavailability of other devices such as
plotters under APLSV. Currently, a device independent plotting system
[GUJA72, GUJA76] is available for FORTRAN users in the Computing Centre

of the University of New Brunswick. The package takes in two dimensional plotting data to plot lines, curves and text strings. Special needs such as line styling are handled by special routines. The interface consists of a FORTRAN program which is submitted through the RJE/RJO facility of APLSV [GUJA75] on behalf of the user of the $\phi SCUBA$ system. The input to this FORTRAN program is a plot file constructed out of the device co-ordinates determined at the time of image construction. This plot file is prepared dynamically as the image is displayed on the view surface of the $\phi SCUBA$ system. A simple sequential file structure is chosen for this plot file which is constructed as a TSIO file. The details of this file structure are reported elsewhere [NAGE79].

The user intending to get a hardcopy has to first initialize his/her plotfile before invoking the $\phi SCUBA$ functions. After displaying the re-quired image, transfer to other devices is handled via an interactive routine TRANSFER which allows the user to specify the device, size of plot, etc. and prepares a remote job to be submitted to the system 370 through RJE/RJO facility [GUJA75]. The output from this job will be the user's hardcopy.

A complete summary of the $\phi SCUBA$ functions is given in Appendix II.

## IV. DEVIATIONS FROM CORE SYSTEM:

One important deviation from the CORE specifications is the facility to retain the object in the world coordinate system. There are two reasons for this. Firstly, this feature conserves the workspace. Secondly, although the image retaining facility allows one to keep

several images of the same object, the advantage of retaining one object definition and viewing it under different viewing conditions interactively seems to be more desirable. Thus three functions, LOAD, STORE and ERASE, are written to work with objects; a residence flag in the segment directory entry indicates the presence or absence of the object in the active workspace. One other advantage of this facility is that viewing of two or more different objects under the same viewing setup is possible.

The file structure for objects is similar to that for images. Line definitions and/or text definitions and/or marker definitions are all stored as sequential real numbers in fixed blocked records. An identification code precedes each set (lines, text or markers) along with the number of entries. Specific format and its size, etc. are discussed elsewhere [NAGE79] in detail.

In addition to the object retaining facility, the hardcopy interface can also be considered as a deviation from CORE specifications. However, this interface is highly modular and can be considered as an auxiliary addition which is easily identifiable and removable. Finally, the interactive nature of the $\phi SCUBA$ system does in a way reflect a slight deviation from the rigorous specifications of the Core System; however, this is a welcome enhancement provided by the base language APL.

## V.  EXPERIMENTAL RESULTS:

In this section, several displays created using the $\phi SCUBA$ system are given.  The world coordinate definitions of the displays created in Figures 4 to 6 are given in Appendix III.

Figure 4 shows a three point perspective view of a garage.  Thr point perspective is obtained by having a view plane which intersects all the three principal axes in the world coordinate system.  The view reference point is the right bottom corner and the center of projection is located at the roof level and is at a fair distance from the garage.

VRP=144 0 120
VPN=0.5 ¯0.25 1
VUP=0 1 0
DIR=50 100 ¯100
TYP=PERSPECTIVE
WIN=¯20 200 0 150
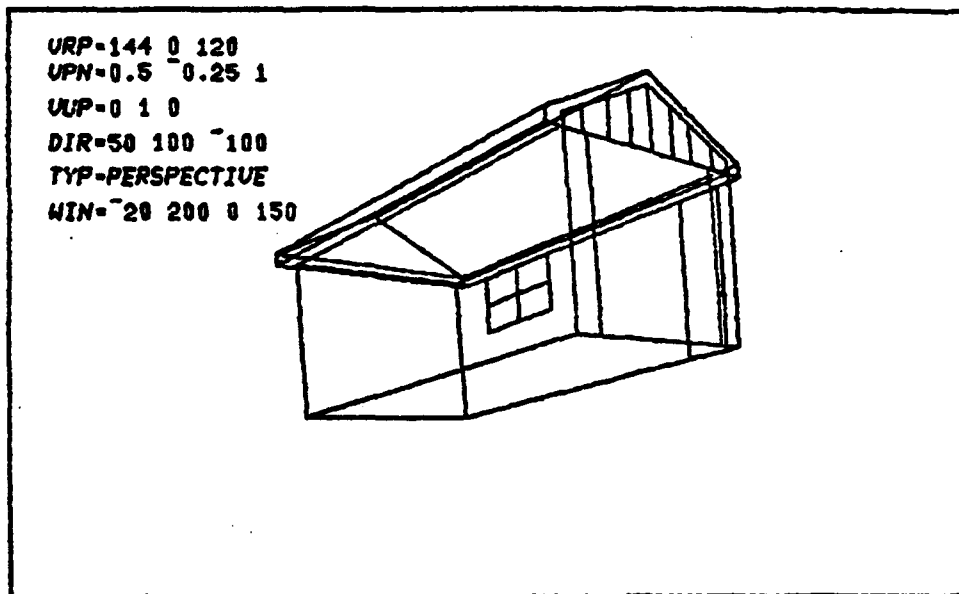
Figure 4:  A Perspective View of a Garage

THIS
PAGE
IS
MISSING
IN
ORIGINAL
DOCUMENT

14

```
VRP=172 0 50
VPN=0 0 1
VUP=0 1 0
DIR=0.577 0.333 ¯1
TYP=OBLIQUE
WIN=¯20 200 0 150
```
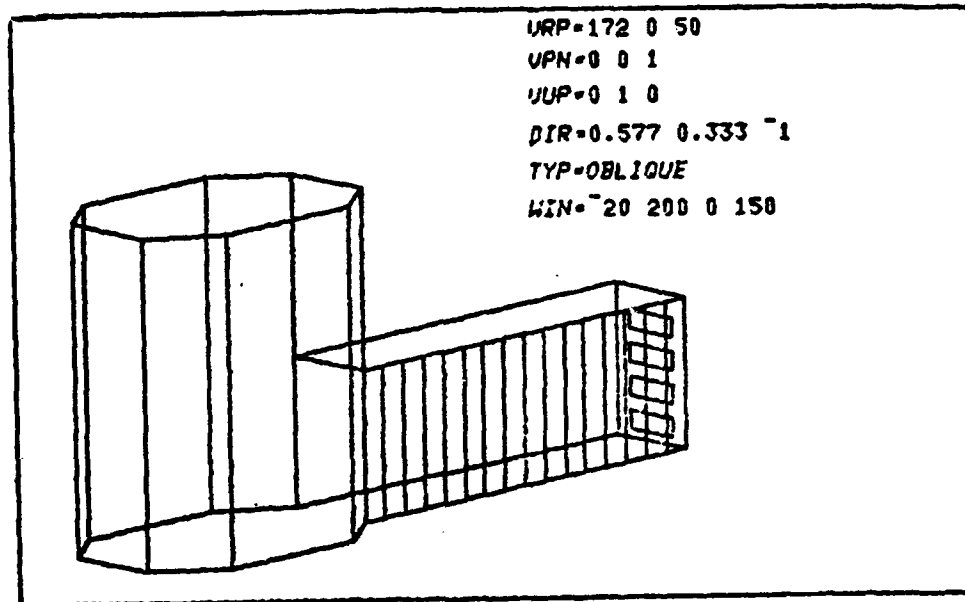
Figure 6:   An Oblique View of a Building
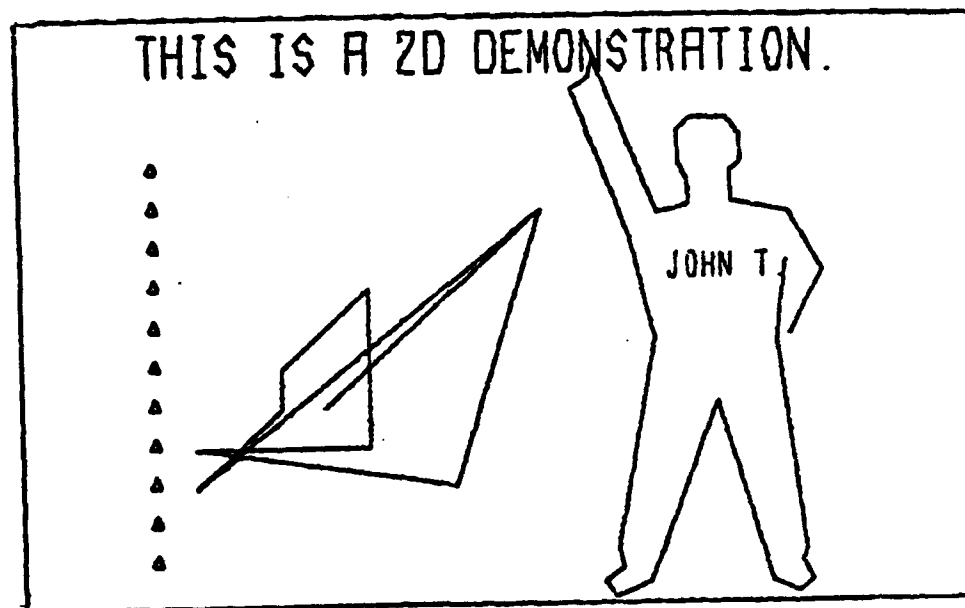


Figure 7:   A 2D Demonstration

Figure 7 shows a two dimensional example. The size and spacing attributes for the two text strings are set by a call to SETPRAT function. The markers appearing on the left are plotted through MARKREL2 called in a loop. The outline of the man is constructed as a polyline through the invocation of a function POLYREL2. The set of lines shown in the figure is just an arbitrary set drawn through the POLYABS2 function. The position of the man is set through the MOVEABS2 function call.

An example of an application program using the $\phi SCUBA$ functions to plot curves is shown in Figure 8. The problem is to create a plot showing two simple trigonometric functions. An auxillary APL function STEP makes up intervals for the plots and the functions SIN and COS evaluate the respective trigonometric functions. The two curves are realized in two different ways. One is plotted as a series of two dimensional markers and the other is "pictured" as a two dimensional object. The function is written using the primitives and is enclosed in a segment for the purposes of adhering to CORE specifications. The results are given in Figure 9.

```
        ∇ PLOT2D;A;B;C;D;E;I              ∇ Z←STEP X
[1]     CREATE                    [1]    Z←X[1]+0,X[3]×
[2]     A← 61 1 ρSTEP 0,(○2),○÷30          ι⌈(X[2]-X[1])÷X[3]
[3]     B←(SIN A)+(SIN 2×A)        ∇
[4]     C←(SIN A)+(COS 2×A)
[5]     D← 61 2 ρA,B
[6]     E← 61 2 ρA,C                      ∇ X←SIN Y
[7]     I←1                       [1]    X←1○Y
[8]     LOOP:'*' MARKABS2 D[I;]    ∇
[9]     →(61≥I←I+1)/LOOP
[10]    MOVEABS2 E[1;]
[11]    POLYABS2(1 0)↓E                   ∇ X←COS Y
[12]    PICTURE                   [1]    X←2○Y
[13]    ⎕←HOME                    ∇
[14]    CLOSE    ∇
```

Figure 8: Program for Plotting Trigonometric Curves
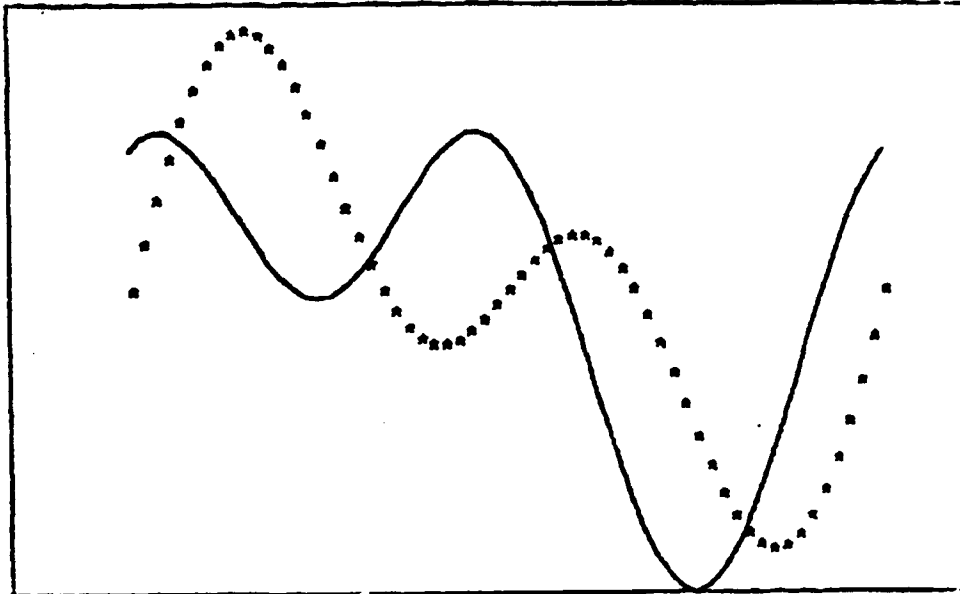
Figure 9:  2D Curve Plotting

Typical segment operations using the Garage example (which appeared
in Figure 4) are illustrated in Figure 10.

```
CREATE
ENTER NAME OF THE SEGMENT [HIT RETURN IF UNNAMED]: SEGMENT1
.....
CLOSE
OPEN
ENTER NAME OF THE SEGMENT [HIT RETURN IF CURRENT]: (Return)
SETSGMT
.....
CLOSE
* IMAGE OF THE CURRENT SEGMENT RETAINED *
STORE
* WORLD DEFINITIONS OF THE SEGMENT SEGMENT1 HAS BEEN RETAINED *
LOAD
ENTER NAME OF THE SEGMENT: SEGMENT2
.....
DESTROY
ENTER NAME OF THE SEGMENT: SEGMENT3
* * * ERROR -- NAMED SEGMENT DOES NOT EXIST * * *
COPY
ENTER NAME OF THE SEGMENT:
.....
```

Figure 10: Illustration of a Few Segment Operations

VI.  <u>CONCLUSIONS</u>:

A graphics system capable of meeting the specifications for a Level
2 (Buffered) Core System, in an APL operating environment, has been
achieved.  The application programmer being an APL user has the advan-
tages of a simple and fast conversational system for the generation and
maintenance of displays. Highly modular in nature, the implementation pro-
vides both two and three dimensional capabilities and supports full seg-
mentation operations.  Provisions are made  to obtain various perspective,
isometric and oblique views as well as side, front and top views commonly
encountered in engineering applications.  Knowledge of APL at an advanced
level, while a distinct advantage on the part of the user, is not manda-
tory for creating and modifying pictures in the ϕSCUBA system.  Experi-
ments with the system so far indicate  that the interactive handling of
setting of attributes, etc. is both efficient and fast.  One of the seri-
ous problems found so far has been that of workspace overflow in cases
of complex objects or multiple segments.  The system itself is still under
development and as such has to be further tested and modified.  Neverthe-
less one can feel the advantages of following GSPC methodology in design-
ing a graphics system.  Further, the task of a graphics application pro-
grammer is simplified when equipped with the functional capabilities of
a Core System.

## REFERENCES

[CARL78]    I. Carlbom, J. Paciorek
            "Planar Geometric Projections and Viewing Transformations".
            ACM Computing Surveys, Vol.10, No.4, pp.465-502, Dec.1978.

[GUJA72]    U.G. Gujar
            "Computer Plotting".
            Computing Center, Univ. of New Brunswick, Fredericton, N.B.
            Canada, Oct.1972.

[GUJA75]    U.G. Gujar
            "Remote Job Entry and Output Through APL".
            APL75: Conference Proceedings, pp.148-157, Pisa, 1975.

[GUJA76]    U.G. Gujar
            "A Device Independent Computer Plotting System".
            ACM Symposium on Graphics Languages
            SIGPLAN Notices, Vol.11, No.6, June 1976
            Computer Graphics, Vol.10, No.1, pp.85-100, Spring 1976.

[GSPC77]    "Status Report of the Graphics Standards Planning Committee".
            Computer Graphics, Vol.11, No.3, Fall 1977.

[NAGE79]    A.R. Nagesh
            "$\phi SCUBA$ - A Buffered Core Graphics System".
            Masters Thesis (to be submitted), Univ. of New Brunswick,
            Fredericton, N.B., Canada, 1979.

[NEWM73]    W.M. Newman, R.F. Sproull
            "Principles of Interactive Computer Graphics"
            McGraw-Hill, 1973.

[NEWM78]    W.M. Newman, A. van Dam
            "Recent Efforts Toward Graphics Standardization".
            ACM Computing Surveys, Vol.10, No.4, pp.365-380, Dec.1978.

[PUK77]     R.F. Puk
            "General Clipping on an Oblique Viewing Frustrum".
            SIGGRAPH '77 Proceedings.
            Computer Graphics, Vol.11, No.2, pp.229-235, Summer 1977.

[ROGE76]    D.F. Rogers, J.A. Adams
            "Mathematical Elements for Computer Graphics".
            McGraw-Hill, 1976.

[UNB74]     "APL Public Library #1 - TSIO Documentation".
            Computing Center, Univ. of New Brunswick, Fredericton, N.B.
            Canada, 1974.

## APPENDIX I

TYPICAL USER SESSION:

A typical conversation with the $\phi SCUBA$ system is given in this appendix. The user input is given in the APL font.

```
      )LOAD NEWTHESIS
SAVED 17.20.15 05 22/79
THIS IS THE VERSION WITHOUT EXAMPLES AND WITHOUT HARDCOPY
      )ERASE CORE4
      )COPY THESIS GARAGE
SAVED 14.47.12 05/18/79
      COREINIT
      SETVIEW     ↔ Viewing parameters are set
VRP:
144 0 120
VPN:
0.5 ‾0.25 1
VPD:
0
TYP:
PERSPECTIVE
DIR:
50 100 ‾100
VUP:
0 1 0
WIN:
‾20 200 0 150
FBD:
‾50 400
NDC:
VPT 0.5 1.0 0.3 0.7/
      CREATE
ENTER NAME OF THE SEGMENT : [HIT RETURN IF UNNAMED] GARAGE
* SEGMENT DIRECTORY SAVED ON TSIO STORAGE *
      LISTSEG
```

| SEGMENT NAME | VISIBILITY | STATUS | RESIDENCE |
|---|---|---|---|
| GARAGE | 1 | 0 | 1 |

VISIBILITY - 1 FOR VISIBLE, 0 FOR INVISIBLE.
   STATUS - 1 FOR RETAINED, 0 FOR NONRETAINED.
 RESIDENCE - 1 FOR RESIDENT, 0 FOR NONRESIDENT.

```
    GARAGE ↔ Output primitives invoked through this function call
    PICTURE ↔ Fig. 4 will be displayed on the screen
      CLOSE
    SETSGMT
CST : TYPE R FOR RETAINING OR
      N OR RETURN OTHERWISE.    R
      OPEN ↔ Segment will be reopened for changes.
ENTER NAME OF THE SEGMENT:  [HIT RETURN IF CURRENT] (Return)
    SETPRAT ↔ New primitive attributes are set.
CIN:
HELP
```

THE FOLLOWING GIVES THE PRIMITIVE ATTRIBUTES THAT CAN BE SET IN ORDER TO VIEW THE SEGMENT WHICH IS CURRENTLY OPEN.  IF NOT INITIALIZED, DEFAULT VALUES WILL BE USED FOR DISPLAYING.

| NAME | KEY | LENGTH | SPECIFICATION | DEFAULT |
|------|-----|--------|---------------|---------|
| CURRENT INTENSITY | CIN | 1 | ABSOLUTE, 0 FOR DIMMED.<br>1 FOR BRIGHT. | 1 |
| CURRENT LINE STYLE | CLS | 1 | ABSOLUTE, 1 FOR FULL LINE.<br>2 FOR DOTTED LINE.<br>3 FOR DOT DASHED.<br>4 FOR SHORT DASHED.<br>5 FOR LONG DASHED. | 1 |
| CURRENT LINE WIDTH | CLW | 1 | ABSOLUTE, 1 FOR NORMAL.<br>2 FOR DOUBLE.<br>3 FOR TRIPLE.<br>4 FOR QUADRUPLE. | 1 |
| CURRENT CHAR. SIZE | CCS | 2 | ABSOLUTE, HEIGHT IN Y-UNITS<br>BY WIDTH IN X-UNITS. | [0,0] |
| CURRENT CHAR. SPACE | CSP | 2 | ABSOLUTE, X-UNITS ALONG WIDTH<br>AND Y-UNITS ALONG HEIGHT. | [0,0] |

PROBABLY YOU CAN CHOOSE AND SET YOUR ATTRIBUTES NOW.  HAPPY VIEWING!

```
CIN:
1
CLS:
3
CLW:
2
CCS:
10 8
CSP:
25 0
     INQUIRE
ENTER NAME OR ABBREVIATION : SEG
CURRENT SEGMENT : GARAGE
     INQUIRE
ENTER NAME OR ABBREVIATION : TYPE
CURRENT SEGMENT TYPE : RETAINED
     MOVEABS3 ¯20 ¯20 650
     TEXT 'GARAGE'
     PICTURE    ↔   Generates a display with a different line style (see below)
     WRITE      ↔   Text will be displayed.
     CLOSE
```

```
* IMAGE OF THE CURRENT SEGMENT HAS BEEN RETAINED *
      STORE
ENTER NAME OF THE SEGMENT : [HIT RETURN IF CURRENT] GARAGE
* WORLD DEFINITIONS OF THE SEGMENT GARAGE HAS BEEN RETAINED *
      DISPLAY
ENTER NAME OF THE SEGMENT : [HIT RETURN IF CURRENT] GARAGE
ENTER ONE OPTION : [ALL/PO/TO/MO/PT/PM/TM] ALL
```

* * * SEGMENT HAS NO MARKERS * * *
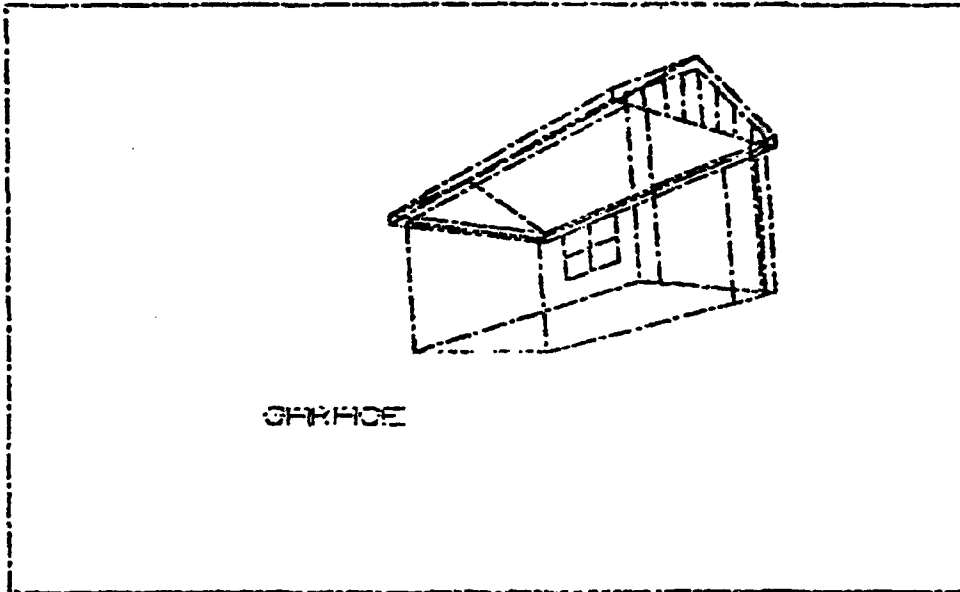


```
      RENAME
ENTER NAME OF THE SEGMENT : GARAGE
ENTER NEW NAME : HOUSE
* SEGMENT DIRECTORY SAVED ON TSIO STORAGE *
* SEGMENT GARAGE HAS BEEN RENAMED AS HOUSE. *
      OPEN
ENTER NAME OF THE SEGMENT : [HIT RETURN IF CURRENT] HOUSE
* * *     NAMED SEGMENT NOT IN WORK SPACE     * * *
* * * LOAD THE SEGMENT BEFORE OPENING AGAIN * * *
      LOAD
ENTER NAME OF THE SEGMENT : HOUSE
* YOUR SEGMENT HAS BEEN LOADED *
      OPEN

      .
      .
      .
      etc.
```

# APPENDIX II

## LIST OF ΦSCUBA FUNCTIONS:

A complete list of ΦSCUBA functions is included in this section. The functions contained and explained in the GSPC77 report are identified with a '+'. Additional functions which may be of direct use to the ΦSCUBA user are identified with '*'; explanation of these follows the list. The remaining functions are mostly used internally.

)GRP CORE1

| | | | | |
|---|---|---|---|---|
| ADJUST | CLIPLINE | CLIP3D | COS | CWRITE |
| CODE | DECODE+ | DISPLINE | DISPOINT | DRAWABS2+ |
| DRAWABS3+ | DRAWREL2+ | DRAWREL3+ | HOME | INVERT |
| LINE | MARKABS2+ | MARKABS3+ | MARKREL2+ | MARKREL3+ |
| MARKPLOT* | MOVEABS2+ | MOVEABS3+ | MOVEREL2+ | MOVEREL3+ |
| NAMETRANS | NMATRIX | NWRITE | OBLIQUE | ORTHO |
| PARAPED | PICTURE* | POLYABS2+ | POLYABS3+ | POLYREL2+ |
| POLYREL3+ | PROJECT | PYRAMID | RESCALEX | RESCALEY |
| RESETSC1 | RESETSC2 | SCALE | SCALEX | SCALEY |
| SELECT | SETSC | SIN | SWAP | TEXT+ |
| TRANSFORM | VWRITE | WINDOW | WRITE* | |

* MARKPLOT   - Creates and displays the image of the markers described by the marker primitives.

* PICTURE    - Creates and displays the image of the wire frame drawing described by the line primitives.

* WRITE      - Creates and displays the image of the text described by the text primitive.


)GRP CORE2

| | | | | |
|---|---|---|---|---|
| CREATE+ | COPY* | CLOSE+ | DESTROY+ | DESTROYALL+ |
| DISPLAY* | ERASE* | INITDIRY* | LISTSEG* | LOAD* |
| LOADDIRY* | NAME* | OPEN* | PADΔ | RENAME+ |
| SRCHDIR | STORDIRY* | STORE* | | |

* COPY       - Copies the contents of a retained segment's image file into work space.

* DISPLAY    - Displays the image (whole or part) of a segment anytime its image definition is in workspace.

* ERASE      - Deletes the retained object definition file of a segment from the system.

* LISTSEG    - Displays the segment directory to the user.

* LOAD       - Loads the retained object definition file of a segment onto the workspace.

*LOADDIRY    - Loads the retained segment directory to the workspace.

* *NAME* — Names an unnamed segment for the purposes of retaining, if needed.

* *OPEN* — Opens a closed segment for picture additions or other output primitive invocations.

* *STORDIRY* — Retains the current segment directory as a TSIO file.

* *STORE* — Retains the current object definitions in a segment as a TSIO file.

* *INITDIRY* — Initializes the segment directory.


*)GRP CORE3*

| | | | | |
|---|---|---|---|---|
| *CHK2D* | *CHK3D* | *CHKCLOSE* | *CHKOPEN* | *CHKVTV* |
| *COREINIT*[+] | *IGLTB* | *INQUIRE** | *PEEL* | *QQP* |
| *SETCTRL** | *SETLEFT*[+] | *SETREST* | *SETRIGHT*[+] | *SETSGMT** |
| *SETVIEW** | *SETPRAT** | *SETVISB*[+] | *TYPDET* | |

* *INQUIRE* — All the possible inquiries such as values of primitive attributes, viewing parameters, segment attributes, etc. are made through this function.

* *SETCTRL* — Sets the clipping options.

* *SETSGMT* — Sets the segment type.

* *SETVIEW* — Sets the viewing parameters.

* *SETPRAT* — Sets the values of primitive attributes.


*)GRP CORE4*

| | | | | |
|---|---|---|---|---|
| *ROTGEN3D** | *ROTATE3D** | *REFLECT3D** | *SCALE3D** | *SHEAR3D** |
| *TRANS3D** | | | | |

* *ROTGEN3D* — Rotates the object about an arbitrary axis.

* *ROTATE3D* — Rotates the object about a coordinate axis.

* *REFLECT3D*— Generates a reflection of the object about any principal plane.

* *SCALE3D* — Scales the object in any or all directions.

* *SHEAR3D* — Produces a shearing of the object in required directions.

* *TRANS3D* — Translates the object to a required point in world coordinate space.


*)GRP COREV*

| | | | | |
|---|---|---|---|---|
| *BOUNDS* | *CMNT* | *CTRLLIST* | *CURPOS* | *CURSEG* |
| *DEFGLBS* | *FLAG2D* | *FLAG3D* | *INCHAR* | *LRFLAG* |
| *OPENFLG* | *PRATLIST* | *PREGLBS* | *ROWKEY* | *SC* |
| *SCRNDIST* | *SGMTDIRY* | *SYM* | *SYSGLBS* | *SCSET* |
| *VIEWLIST* | *WC* | *GWCDEF* | *GMKDEF* | *GTXDEF* |
| *GLIMGE* | *GMIMGE* | *OLDSC* | | |

)GRP HARDCOPY

| DROPOS | HCOPYINIT* | JOBSECU | NUMTOCHR | PLOTFILE |
|--------|-----------|---------|----------|----------|
| PRINT  | PURGE     | RJERJO  | TRANSFER* | FILE    |
| ACT    | FOR       | JB      | JOBSECU  | LL       |
| N      | PF        | PN      | PRT      | STRT     |
| CT     |           |         |          |          |

* HCOPYINIT - Initializes the TSIO file to record 'pen movements'.

* TRANSFER  - Transfers the screen display to the required hardcopy
             unit.


)GRP TSIO

| CHK | DELETE | RELEASE | TRY | QLE |
|-----|--------|---------|-----|-----|
| PID | Q      |         |     |     |


)GRP TEKGRP

| CONVERT | CORD   | ERASE | L      | LT   |
|---------|--------|-------|--------|------|
| P       | SCREEN | SIZE  | WRITE  | CH   |
| H       | R      | S     | XYDIMN | ALPH |
| BS      | CSC    | CSV   | CUR    | GRF  |
| HXY     | OM     | RS    | US     | W    |
| XL      | YL     |       |        |      |

## APPENDIX III

The world coordinate definitions of the examples used in constructing Figures 4, 5 and 6 are given below.  A left handed system is assumed.

*GARAGE* (80 × 4 array)

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 144 | 84 | 390 | 0 | 144 | 48 | 318 |
| 0 | -12 | 84 | 120 | 1 | 0 | 84 | 390 | 1 | 144 | 80 | 318 |
| 1 | -12 | 90 | 120 | 0 | 144 | 0 | 390 | 0 | 144 | 64 | 354 |
| 1 | 72 | 120 | 120 | 1 | 0 | 0 | 390 | 1 | 144 | 64 | 282 |
| 1 | 156 | 90 | 120 | 0 | 144 | 84 | 390 | 0 | -12 | 90 | 396 |
| 1 | 156 | 84 | 120 | 1 | 144 | 0 | 390 | 1 | -12 | 90 | 120 |
| 1 | 144 | 84 | 120 | 0 | 0 | 84 | 390 | 0 | -12 | 84 | 396 |
| 1 | 144 | 90 | 120 | 1 | 0 | 0 | 390 | 1 | -12 | 84 | 120 |
| 1 | 72 | 114 | 120 | 0 | 72 | 120 | 120 | 0 | 0 | 84 | 390 |
| 1 | 0 | 90 | 120 | 1 | 72 | 120 | 396 | 1 | 0 | 84 | 120 |
| 1 | 0 | 0 | 120 | 0 | 156 | 90 | 120 | 0 | 0 | 0 | 390 |
| 1 | 144 | 0 | 120 | 1 | 156 | 90 | 396 | 1 | 0 | 0 | 120 |
| 1 | 144 | 84 | 120 | 0 | 156 | 84 | 120 | 0 | 15 | 95 | 120 |
| 1 | -12 | 84 | 120 | 1 | 156 | 84 | 396 | 1 | 15 | 84 | 120 |
| 0 | 18 | 84 | 120 | 0 | 144 | 0 | 120 | 0 | 36 | 102 | 120 |
| 1 | 18 | 0 | 120 | 1 | 144 | 0 | 390 | 1 | 36 | 84 | 120 |
| 0 | 126 | 84 | 120 | 0 | 144 | 84 | 390 | 0 | 54 | 108 | 120 |
| 1 | 126 | 0 | 120 | 1 | 144 | 84 | 120 | 1 | 54 | 84 | 120 |
| 0 | 156 | 84 | 396 | 0 | 144 | 0 | 129 | 0 | 72 | 114 | 120 |
| 1 | 156 | 90 | 396 | 1 | 144 | 80 | 129 | 1 | 72 | 84 | 120 |
| 1 | 72 | 120 | 396 | 1 | 144 | 80 | 159 | 0 | 90 | 108 | 120 |
| 1 | -12 | 90 | 396 | 1 | 144 | 0 | 159 | 1 | 90 | 84 | 120 |
| 1 | -12 | 84 | 396 | 0 | 144 | 48 | 282 | 0 | 108 | 102 | 120 |
| 0 | -12 | 90 | 396 | 1 | 144 | 48 | 354 | 1 | 108 | 84 | 120 |
| 1 | 156 | 90 | 396 | 1 | 144 | 80 | 354 | 0 | 129 | 95 | 120 |
| 0 | 156 | 84 | 396 | 1 | 144 | 80 | 282 | 1 | 129 | 84 | 120 |
| 1 | -12 | 84 | 396 | 1 | 144 | 48 | 282 | | | | |

*STAIRS* (42 × 4 array)

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 13 | 60 | 30 | 0 | 13 | 45 | 10 |
| 0 | 0 | 0 | 10 | 1 | 13 | 45 | 30 | 1 | 13 | 45 | 30 |
| 1 | 0 | 60 | 10 | 1 | 26 | 45 | 30 | 0 | 26 | 45 | 30 |
| 1 | 13 | 60 | 10 | 1 | 26 | 30 | 30 | 1 | 26 | 45 | 10 |
| 1 | 13 | 45 | 10 | 1 | 39 | 30 | 30 | 0 | 26 | 30 | 10 |
| 1 | 26 | 45 | 10 | 1 | 39 | 15 | 30 | 1 | 26 | 30 | 30 |
| 1 | 26 | 30 | 10 | 1 | 52 | 15 | 30 | 0 | 39 | 30 | 30 |
| 1 | 39 | 30 | 10 | 1 | 52 | 0 | 30 | 1 | 39 | 30 | 10 |
| 1 | 39 | 15 | 10 | 1 | 0 | 0 | 30 | 0 | 39 | 15 | 10 |
| 1 | 52 | 15 | 10 | 1 | 0 | 0 | 10 | 1 | 39 | 15 | 30 |
| 1 | 52 | 0 | 10 | 0 | 0 | 60 | 10 | 0 | 52 | 15 | 30 |
| 1 | 0 | 0 | 10 | 1 | 0 | 60 | 30 | 1 | 52 | 15 | 10 |
| 0 | 0 | 0 | 30 | 0 | 13 | 60 | 30 | 0 | 52 | 0 | 10 |
| 1 | 0 | 60 | 30 | 1 | 13 | 60 | 10 | 1 | 52 | 0 | 30 |

## BUILDING (101 × 4 array)

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 38 | 0 | 66 | 1 | 142 | 50 | 66 |
| 0 | 50 | 0 | 50 | 0 | 38 | 0 | 96 | 0 | 147 | 0 | 66 |
| 1 | 80 | 0 | 50 | 1 | 38 | 110 | 96 | 1 | 147 | 50 | 66 |
| 1 | 92 | 0 | 66 | 0 | 50 | 110 | 112 | 0 | 152 | 0 | 66 |
| 1 | 172 | 0 | 66 | 1 | 50 | 0 | 112 | 1 | 152 | 50 | 66 |
| 1 | 172 | 0 | 96 | 0 | 80 | 0 | 112 | 0 | 157 | 0 | 66 |
| 1 | 92 | 0 | 96 | 1 | 80 | 110 | 112 | 1 | 157 | 50 | 66 |
| 1 | 80 | 0 | 112 | 0 | 92 | 110 | 96 | 0 | 162 | 0 | 66 |
| 1 | 50 | 0 | 112 | 1 | 92 | 0 | 96 | 1 | 162 | 50 | 66 |
| 1 | 38 | 0 | 96 | 0 | 92 | 0 | 66 | 0 | 167 | 0 | 66 |
| 1 | 38 | 0 | 66 | 1 | 92 | 110 | 66 | 1 | 167 | 50 | 66 |
| 1 | 50 | 0 | 50 | 0 | 80 | 110 | 50 | 0 | 172 | 0 | 66 |
| 0 | 50 | 110 | 50 | 1 | 80 | 0 | 50 | 0 | 172 | 3 | 72 |
| 1 | 80 | 110 | 50 | 0 | 92 | 50 | 66 | 1 | 172 | 9 | 72 |
| 1 | 92 | 110 | 66 | 1 | 92 | 50 | 96 | 1 | 172 | 9 | 90 |
| 1 | 92 | 110 | 96 | 0 | 97 | 0 | 66 | 1 | 172 | 3 | 90 |
| 1 | 80 | 110 | 112 | 1 | 97 | 50 | 66 | 1 | 172 | 3 | 72 |
| 1 | 50 | 110 | 112 | 0 | 102 | 0 | 66 | 0 | 172 | 14 | 72 |
| 1 | 38 | 110 | 96 | 1 | 102 | 50 | 66 | 1 | 172 | 20 | 72 |
| 1 | 38 | 110 | 66 | 0 | 107 | 0 | 66 | 1 | 172 | 20 | 90 |
| 1 | 50 | 110 | 50 | 1 | 107 | 50 | 66 | 1 | 172 | 14 | 90 |
| 0 | 92 | 110 | 66 | 0 | 112 | 0 | 66 | 1 | 172 | 14 | 72 |
| 1 | 92 | 50 | 66 | 1 | 112 | 50 | 66 | 0 | 172 | 25 | 72 |
| 1 | 172 | 50 | 66 | 0 | 117 | 0 | 66 | 1 | 172 | 31 | 72 |
| 1 | 172 | 50 | 96 | 1 | 117 | 50 | 66 | 1 | 172 | 31 | 90 |
| 1 | 92 | 50 | 96 | 0 | 122 | 0 | 66 | 1 | 172 | 25 | 90 |
| 1 | 92 | 110 | 96 | 1 | 122 | 50 | 66 | 1 | 172 | 25 | 72 |
| 0 | 172 | 50 | 66 | 0 | 127 | 0 | 66 | 0 | 172 | 36 | 72 |
| 1 | 172 | 0 | 66 | 1 | 127 | 50 | 66 | 1 | 172 | 42 | 72 |
| 0 | 172 | 0 | 96 | 0 | 132 | 0 | 66 | 1 | 172 | 42 | 90 |
| 1 | 172 | 50 | 96 | 1 | 132 | 50 | 66 | 1 | 172 | 36 | 90 |
| 0 | 50 | 0 | 50 | 0 | 137 | 0 | 66 | 1 | 172 | 36 | 72 |
| 1 | 50 | 110 | 50 | 1 | 137 | 50 | 66 | 0 | 172 | 47 | 72 |
| 0 | 38 | 110 | 66 | 0 | 142 | 0 | 66 | | | | |

# TECHNICAL REPORTS

## SCHOOL OF COMPUTER SCIENCE

| Number | Date | Author | Title |
|--------|------|--------|-------|
| TR74-001 | Feb. 1974 | L. F. Johnson | A Search Algorithm for the Simple Cycles of a Directed Graph |
| TR74-002 | Oct. 1974 | W. D. Wasson<br>R. McIssaac | A New Spanning Tree Algorithm |
| TR74-003 | Oct. 1974 | U. G. Gujar | Remote Job Entry and Output Through APL |
| TR75-004 | Apr. 1975 | U. G. Gujar | Subroutines with Variable Number of Arguments |
| TR75-005 | July 1975 | L. E. Garey | Block Methods for Nonlinear Volterra Integral Equations |
| TR75-006 | Aug. 1975 | D. M. Fellows | Comments on "A General Fortran Emulator for IBM 360/370 Random Number Generator 'RANDU'" |
| TR75-007 | Aug. 1975 | L. E. Garey<br>M. LeBlanc | Quadrature Formulae for Functions of Two Variables and Applications |
| TR75-008 | Sept.1975 | L. F. Johnson | Determining Cliques of a Graph |
| TR75-009 | Oct. 1975 | D. M. Miller | An Algorithm for Determining The Chromatic Number of a Graph |
| TR76-010 | Jan. 1976 | L. E. Garey | Step by Step Methods for the Numerical Solution of Volterra Integro-Differential Equations |

## TECHNICAL REPORTS (continued)

| Number | Date | Author | Title |
|---|---|---|---|
| TR76-011 | Jan. 1976 | Uday G. Gujar | A Device Independent Computer Plotting System |
| TR76-012 | Mar. 1976 | Patrick P. Emin | A Partition Monitor for Fast-Batch-Processing with Limited Execution   (Fable) |
| TR76-013 | Dec. 1976 | Uday G. Gujar J. A. Fitzgerald | A Driver for Raster-Like Plotting Devices |
| TR77-014 | Jan. 1977 | Uday G. Gujar David M. Fellows | Automatic Job Scheduling in HASP |
| TR79-015 | May 1979 | Uday G. Gujar John M. DeDourek Marion E. McIntyre | A Method for Designing a Lexical Analyzer |
| TR79-016 | June 1979 | Uday G. Gujar Aragam R. Nagesh | $\phi SCUBA$ A Buffered Core Graphics System |