

☐ UPDATE DESIGN  
☐ UPDATE Disc  
☐ UPDATE ~~PROTOTYP~~ Source

SPECIFICATION: 3D DISPLAY OF PLANET FOR STARQUEST

11-28-83

-----I N D E X-----

1.0 DESIGN REQUIREMENTS

2.0 FUNDAMENTAL DECISIONS

2.1 GLOBE SPECIFICATION

2.2 MERCATOR MAP SPECIFICATION

2.3 LANDMASS BOUNDARY EXTRACTION FROM SURVEY SQUARE DATA

2.4 LANDMASS REPRESENTATION

2.5 GRAPHICS DISPLAY BUFFER

2.6 DISPLAY SCREEN WINDOW COORDINATES

3.0 FUNCTIONAL DESCRIPTION

3.1 LANDMASS DISPLAY LIST CONSTRUCTION

3.2 PROJECTION OF GLOBE FACETS INTO VIEWING PLANE

3.3 DISPLAY LIST PLOTTING

3.4 GRAPHICS BUFFER TO SCREEN

3.5 FILL GRAPHICS BUFFER

3.6 CLIP A LINE SEGMENT AGAINST THE GBUFFER BOUNDARIES

3.7 PLOT A LINE IN GBUFFER

3.8 PLOT A POINT IN GBUFFER

3.9 SQUARE ROOT FUNCTION

4.0 DATA STRUCTURES

4.1 GLOBE MODEL DATA

4.1.1 FACET ARRAY

4.1.2 LINE-SEG ARRAY

4.1.3 VERTEX ARRAY

4.1.4 NORMAL ARRAY

4.1.5 VERTEX' ARRAY

4.1.6 LINE-DRAWN ARRAY

4.2 MERCATOR MAP DATA

4.3 FACET TO DISPLAY LIST POINTER TABLE

4.4 DISPLAY LIST

4.5 DISPLAY LIST CONSTRUCTION ARRAY

5.0 LOGICAL DESIGN

5.1 LANDMASS DISPLAY LIST CONSTRUCTION

5.1.1 LANDMASS BOUNDARY LINE ENDPOINT IDENTIFICATION

5.1.2 BOUNDARY ENDPOINT CONNECTION

5.1.2.1 BOUNDARIES CLOSED EXTERNAL TO THE REGION

5.1.2.2 BOUNDARIES CLOSED WITHIN THE REGION

5.1.2.3 GET NEXT POINT IN BOUNDARY LIST

5.2 RENDER THE GLOBE AS VIEWED

5.2.1 BUILDING THE VIEWING TRANSFORMATION

5.2.2 CALCULATING THE PROJECTED ENDPOINTS

5.2.3 SELECTING THE VISIBLE FACETS

5.2.4 CLIPPING THE LINE SEGMENTS TO WINDOW BOUNDARIES

5.2.5 DRAWING THE LINE SEGMENTS

5.3 DISPLAY LIST PLOTTING

5.4 MOVING THE DISPLAY BUFFER TO THE SCREEN BUFFER

6.0 FORTH SOURCE CODE

## 1.0 DESIGN REQUIREMENTS

Design a set of modules that:

- 1) Given a Mercator representaion of terrain on a planet, construct a landmass boundary display list.
- 2) Given a landmass boundary display list, viewpoint coordinates and globe data, display a representation of a 3D globe with visible landmass boundaries drawn.

Optimize for speed, portability and aesthetics.

## 2.0 FUNDAMENTAL DECISIONS

### 2.1 GLOBE SPECIFICATION

The globe shall be a polyhedron, radially semetric about the polar axis with line segments representing latitude and longitude every 30 degrees of arc. This results in 6 latitudes of 12 Facets each for a total of 72 Facets. The southern most and northern most latitudes are formed by triangular Facets with all the rest being parallelograms.

### 2.2 MERCATOR MAP SPECIFICATION

The dimensions of the Mercator projection that allow a one-to-one mapping of area to Facets is 48 by 25 Survey Squares. This results in 72 Regions of 5x5 Survey Squares, with a row of Survey Squares common to each adjoining Region (for inter-Facet connections). Note that the Eastern most Regions must obtain a row of Survey Squares from the Western boundary to overlap.

### 2.3 LANDMASS BOUNDARY EXTRACTION FROM SURVEY SQUARE DATA

Assuming that Survey Squares can contain any one of 16 terrain types in the form of 4 bit codes, code 0000 is reserved for non-landmass terrain.

### 2.4 LANDMASS REPRESENTATION

In the interest of speed, landmasses will be represented by closed boundary lines showing the demarcation between land and non-land. Closure of boundary lines may occur outside of a Facet as well as within. The resolution is 9x9 for each Region/Facet. Boundary line segments are drawn within each Facet using the projected Facet corners as reference points and interpolating to determine the screen coordinates of the end points.

### 2.5 GRAPHICS DISPLAY BUFFER

An area of memory is set aside for picture frame construction called the "GBUFFER". All point plotting is done within the GBUFFER and the complete picture is transferred to the display window in a block memory move. The display window and the GBUFFER buffer have the same dimensions, 72x120 for a total of 2160 bytes.

### 2.6 DISPLAY SCREEN WINDOW COORDINATES

The lower left corner of the display window is located at ??

*upper* ←

(4,8)  
X

### 3.0 FUNCTIONAL DESCRIPTION

#### 3.1 LANDMASS DISPLAY LIST CONTRUCTION

This submodule builds a landmass boundary display list from the Mercator map array given. Extraneous line segment endpoints are discarded (ie. those segment endpoints that can be eliminated without changing the shape or length of the boundary). This list is constructed only once and it is viewpoint independent.

#### 3.2 PROJECTION OF GLOBE FACETS INTO VIEWING PLANE

Given viewpoint parameters in the form of X,Y,Z and magnification, a drawing of the globe is constructed in the graphics buffer (GBUFFER) with only visible surfaces shown.

#### 3.3 DISPLAY LIST PLOTTING

The display list generated in 3.1 is plotted in the GBUFFER for each visible Facet determined in 3.2 using the the projected Facet vertices as orientation points and interpolating between. Line segments are drawn to represent landmass boundaries.

#### 3.4 GRAPHICS BUFFER TO SCREEN

The GBUFFER is moved to the display screen to provide the appearance of instantaneous picture transition between frames.

#### 3.5 FILL GRAPHICS BUFFER

Clears the GBUFFER between frames by setting all pixels to the background color.

#### 3.6 CLIP A LINE SEGMENT AGAINST THE GBUFFER BOUNDARIES

Given any pair of endpoints, return the endpoints of a line segment that falls within the GBUFFER.

#### 3.7 PLOT A LINE IN GBUFFER

Given the endpoints of a line segment within GBUFFER, calculate the x,y coordinates of the pixels that compose that line and plot them.

#### 3.8 PLOT A POINT IN GBUFFER

Writes the last selected color to the pixel at x,y with the lower left corner of the GBUFFER being the origin. Resolution is 160x200 (low res).

#### 3.9 SQUARE ROOT FUNCTION

Given double length (32 bit) number, calculate the square root (16 bit). Used in viewing transformation construction.



#### 4.0 DATA STRUCTURES

##### 4.1 GLOBE MODEL DATA

This data is used to render the polyhedral globe.

##### 4.1.1 FACET ARRAY

The Facets are organized from North to South with latitudinal sets aligned on the same starting longitude. Each Facet contains a list of 4 line segments that form it's edges (for the Facets in the polar latitudes a line of zero length is included, in other words, triangular Facets are considered to be parallelograms with one of the parallel sides of zero length. This allows the landmass plotting algorithm to operate uniformly on all surfaces.)

The line segments are stored in this order:

- 1 - Northern-most latitude line (N)
- 2 - Southern-most latitude line (S)
- 3 - Western-most longitude line (W)
- 4 - Eastern-most longitude line (E)

Each Facet occupies 4 bytes of storage for a total of 288 bytes for this array. The FORTH format is:

```
CREATE FACET
  lnN C, lnS C, lnW C, lnE C, ( Facet # 1)
  lnN C, lnS C, lnW C, lnE C, ( Facet # 2)
  .
  .
  lnN C, lnS C, lnW C, lnE C, ( Facet # 72)
```

...where lnx is the index into the LINE-SEG array.

##### 4.1.2 LINE-SEG ARRAY

Each line segment entry consists of an index to each of the endpoints in the VERTEX array. Each LINE-SEG element occupies 2 bytes for a total of 268 bytes (134 line segments). The endpoints for latitudinal lines are organized West-East; longitudinal lines are organized North-South. The FORTH format is:

```
CREATE LINE-SEG
  pt1 C, pt2 C, ( line segment # 1)
```

...where ptx is the index into the VERTEX array.

#### 4.1.3 VERTEX ARRAY

62  
3  
186

The vertex points of the polyhedron are specified in terms of x,y,z coordinates. Each vertex entry occupies 3 bytes; there are 62 vertices for a total of 378 bytes. The Vertices are organized North-South, West-East with the Y axis ~~extending through the poles. parallel to the z-axis.~~ FORTH format is:

```
CREATE VERTEX
x1C, y1C, z1C,          ( vertex # 1)
```

...where x1,y1,z1 are ~~signed 8~~ bit integers with values ~~-4032 <= x <= 4032~~ ~~(tradeoff made between space and time; extra byte used for sign information to avoid un packing and sign extension each time number used.)~~

#### 4.1.4 NORMAL ARRAY

Immediately following the 62 vertices are 72 points normal to each of the Facets. These normal points are used for determining which Facets are visible from the viewpoint. ~~The storage format is identical to the vertex points and the normal points are appended to the VERTEX array to simplify transformation of globe-related coordinates.~~ The normal points are located in space at a point normal to the North-West vertex of each Facet and stored in the same order as the Facet array. A total of 432 bytes are occupied by the normal points. The FORTH format is:

```
x62 , y62 , z62          ( vertex # 62)
xn1 , yn1 , zn1          ( normal # 1)
.
.
xn72 , yn72 , zn72       ( normal # 72)
```

...where xnx is a signed 16 bit number with values -375 <= x <= 375.

#### 4.1.5 VERTEX' ARRAY

This array contains the vertex and normal points after the viewing transformation has been applied. The format is identical to 4.1.3 & 4.1.4 and starts at the PFA of VERTEX'. The FORTH format for this temporary workspace is:

```
CREATE VERTEX' 804 ALLOT
```

#### 4.1.6 LINE-DRAWN ARRAY

This array is contains a set of byte flags to indicate whether a line segment has been drawn in GBUFFER. It occupies 134 bytes. The FORTH format for this temporary workspace is:

```
CREATE LINE-DRAWN 134 ALLOT
```

10

#### 4.2 MERCATOR MAP DATA

The given information of the Mercator map is stored with two survey squares per byte organized in 25 rows of 24 bytes per row for a total of 600 bytes. The FORTH format is:

CREATE MERCATOR 600 ALLOT

#### 4.3 FACET TO DISPLAY LIST POINTER TABLE

An address pointer table is built to link Facets to their associated landmass display list. If there is no display list associated with a given Facet the address entry contains a 0. There are 72 elements of two bytes each for a total of 144 bytes. The FORTH format is:

CREATE FACET-TO-DISPLAY 144 ALLOT

*Continue here.*

#### 4.4 DISPLAY LIST

*6336 worst case likely 2 rows 1 K*

The landmass boundary display list consists of 72 or fewer lists of varying length. Space occupied by this list in a worst case situation depends on the packing level used: 3456 bytes (unpacked), 1728 bytes (packed). In the interest of speed the unpacked format was chosen. The list associated with a given Facet takes the following format:

x1 C, y1 C, x2 C, y2 C, x3 C, 9 C, x4 C, y4 C, ... 10 C,

...where xn,yn are endpoints of line segments in the Facet coordinate system

9 - corresponds to the plotting command  
"lift pen and move to next point"

10 - is the list terminator command (decimal)

(Note: the Facet coordinate system is a rectangular coordinate system with range of values 0-8 with the origin at the Southwest corner of the Facet.) If a display list exists for a given Facet, the address of x1 for that list is found by looking in the FACET-TO-DISPLAY list. The endpoint list is interpreted to mean: "connect point 1 to point 2, connect point 2 to point 3..." until a "lift pen" or terminator is reached. The FORTH format is:

CREATE DISPLAY-LIST

x1 C, y1 C, x2 C, y2 C, ... 9 C, ... 10 C,

#### 4.5 DISPLAY LIST CONSTRUCTION ARRAY

The array consists of 9 rows of 9 bytes for a total of 81 bytes. This array is used during construction of the display list. The FORTH format is:

CREATE DISARRAY 81 ALLOT

## 5.0 LOGICAL DESIGN

### 5.1 LANDMASS DISPLAY LIST CONSTRUCTION

```
Clear FACET-TO-DISPLAY array
FOR each Region in Mercator
  (1) Identify boundary line segment endpoints (5.1.1)
  Connect endpoints into display list (5.1.2)
  IF top of stack = 9 (do not preserve 9 in test)
    Store address of display list in FACET-TO-DISPLAY
    Store display list in dictionary
  THEN
NEXT
END
```

#### 5.1.1 LANDMASS BOUNDARY LINE ENDPOINT IDENTIFICATION

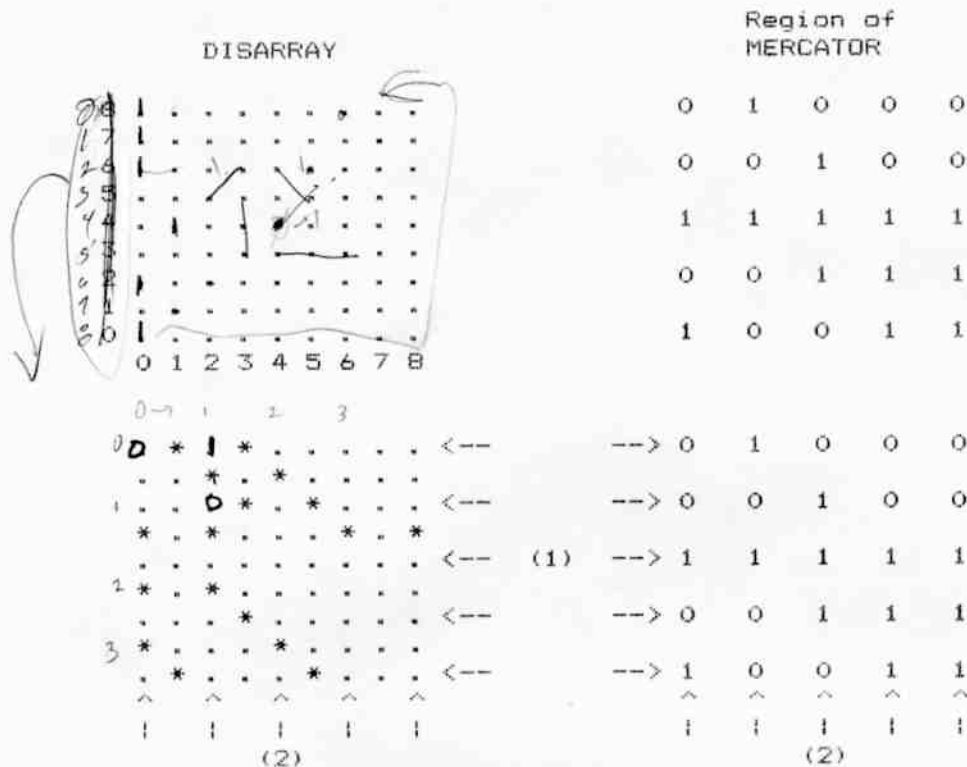
To provide a pleasing display the resolution for each Region is increased from 5x5 to 9x9 to allow a half step between each Survey for boundary connection. This routine identifies the location of all these half-step locations found between land and non-land Survey Squares. Pseudo-code and an example follow:

```
Clear DISARRAY flags
FOR each row in Region (1- see example)
  FOR each column in Region
    IF transition between land and non-land
      Set the DISARRAY flag found between surveys
    THEN
  NEXT column
NEXT row
FOR each column in Region (2- see example)
  FOR each row in Region
    IF transition between land and non-land
      Set the DISARRAY flag found between surveys
    THEN
  NEXT row
NEXT column
```

( SEE EXAMPLE ON FOLLOWING PAGE )



### 5.1.1 EXAMPLE



### 5.1.2 BOUNDARY ENDPOINT CONNECTION

This routine takes the set of endpoints identified in 5.1.1 and extracts the boundary segment lists that are formed by the connection of the points. The input is DISARRAY; the output is left on the stack in the display list format.

#### 5.1.2.1 BOUNDARIES CLOSED EXTERNAL TO THE REGION

Push 10 into stack as list terminator

FOR each edge coordinate in DISARRAY

IF DISARRAY flag set

Clear the DISARRAY flag

Push endpoint coordinates into stack

BEGIN

Get next point in boundary (see 5.1.2.3)

Clear the DISARRAY flag

Push endpoint coord into stack

IF 2nd stack item is co-linear with the 1st & 3rd

Drop 2nd stack item

THEN

Is current point is on boundary ?

UNTIL

Push a 9 into the stack (lift pen)

THEN

NEXT edge coordinate



### 5.1.2.2 BOUNDARIES CLOSED WITHIN THE REGION

```

FOR each interior coordinate in DISARRAY
  IF DISARRAY flag set
    Set the DISARRAY flag to -1 to indicate start
    Push endpoint coordinates into stack
  BEGIN
    Get next point in boundary (see 5.1.2.3)
    IF DISARRAY flag 1
      Clear the DISARRAY flag
    THEN
      Push endpoint coordinate into stack
      IF 2nd stack item is co-linear with the 1st & 3rd
        Drop 2nd stack item
      THEN
        Is DISARRAY flag set?
      UNTIL
        Clear the DISARRAY flag
        Push a 9 into the stack (lift pen)
    THEN
  NEXT interior coordinate

```

### 5.1.2.3 GET NEXT POINT IN BOUNDARY LIST

From any given point in the DISARRAY workspace a search is made for the next location of the boundary. The search order is adjacent diagonal connections first and if not found try either vertical or horizontal location at a distance of two units. The choice of either vertical or horizontal search in the fallback case is dependent on the location of the current endpoint. (see example and pseudo-code).

#### EXAMPLE OF SEARCH SEQUENCE

. . .6 . .	Adjacent diagonals
. .3   4. .	1 - Southeast
\   /	2 - Southwest
6*<=====*.5	3 - Northwest
/   \	4 - Northeast
. .2   1. .	2 unit distant horiz/vertical
. . .5 . .	5 - East or South
	6 - West or North

The search direction in the fallback case is in the direction that is not in line with any of the even numbered coordinates (ie. if the current endpoint illustrated is 6,3 the fallback search direction is East-West, the next point being found at 4,3).

### 5.1.2.3 GET NEXT POINT IN BOUNDARY LIST (con't)

```

Clear FOUND flag
IF current Y<8 AND X<8
    IF SE adjacent set
        Push coordinate & set FOUND
    THEN
ELSE
    IF current Y<8 AND X>0
        IF SW adjacent set
            Push coordinate & set FOUND
        THEN
    ELSE
        IF current Y>8 AND X>0
            IF NW adjacent set
                Push coordinate & set FOUND
            THEN
        ELSE
            IF current Y>8 AND X<8
                IF NE adjacent set
                    Push coordinate & set FOUND
                THEN
            THEN
        THEN
    THEN
THEN
IF FOUND not set
    IF current X is even
        IF current X<7
            IF East, 2 units distant set
                Push coordinate & set FOUND
            ELSE
                Push coordinate of West, 2 units distant
                & set FOUND
            THEN
        ELSE Push coordinate of West, 2 units distant
            & set FOUND
        THEN
    ELSE
        IF current Y<7
            IF South, 2 units distant set
                Push coordinates & set FOUND
            ELSE
                Push coordinate of North, 2 units distant
                & set FOUND
            THEN
        ELSE Push coordinate of North, 2 units distant
            & set FOUND
        THEN
    THEN
THEN

```

## 5.2 RENDER THE GLOBE AS VIEWED

A viewing transformation is constructed which when applied to the vertex coordinates of the globe produces a set of points on the viewing plane. The transformation is applied to the normal points associated with each Facet to determine which ones are visible. Lines are drawn for the surface edges with a flag table maintained to avoid duplicate line drawing (lines are clipped to viewing window boundaries). A complete pseudo-code flow of the transformation and display follows:

```
Clear GBUFFER
Clear the LINES-DRAWN array
Set the x" array values to 9999
Build the viewing transformation (see 5.2.1)
FOR each vertex point and each normal point
    Calculate z' (see 5.2.2)
NEXT
FOR each Facet
    IF Facet is visible (see 5.2.3)
        FOR each vertex point
            IF x"=9999
                Calculate x",y" & save in array (see 5.2.2)
            THEN
                NEXT
        FOR each line segment
            IF line not drawn in GBUFFER
                Draw the line segment
            THEN
                NEXT
        IF there is a landmass boundary on this Facet
            BEGIN
                Draw boundary
            End?
            UNTIL
        THEN
            THEN
        NEXT
Move GBUFFER to the display
END
```



### 5.2.1 BUILDING THE VIEWING TRANSFORMATION

The coordinate system of the world coordinates is assumed to be as diagrammed where the axis identifier is in the positive direction.



The viewing coordinate system is constructed to view the origin of the world coordinate system from the view point.



The viewing transformation performs 3 steps:

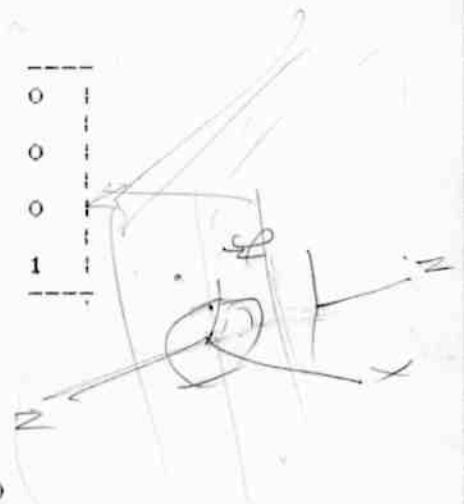
- 1 - Rotate the viewing coordinate system about the  $y'$  axis by  $-a$  degrees, where "a" is the angle that must be traversed to align the  $z'$  axis with the  $y$  axis.
- 2 - Rotate the viewing coordinate system about the  $x'$  axis by  $-b$  degrees, where "b" is the angle that must be traversed to point the  $z'$  axis at the origin.
- 3 - Flip the  $z'$  axis to be positive towards the origin.

The resultant transformation matrix is:

$$\begin{array}{c}
 \text{---} \\
 \begin{array}{ccc|ccc}
 \vdots & \cos a & \sin a \sin b & \sin a \cos b & 0 & \vdots \\
 \vdots & 0 & \cos b & -\sin b & 0 & \vdots \\
 \vdots & \sin a & -\cos a \sin b & -\cos a \cos b & 0 & \vdots \\
 \vdots & 0 & 0 & h' & 1 & \vdots \\
 \text{---} & & & & \text{---} & 
 \end{array}
 \end{array}$$

where:

$$\begin{array}{l}
 \cos a = z \text{ (view coordinate)} \\
 \sin a = -x \text{ (view coordinate)} \\
 h = \text{SQRT}(z*z + x*x) \\
 h' = \text{SQRT}(h*h + y*y) \\
 \cos b = h/h' \\
 \sin b = y/h' \text{ ( y view coordinate )}
 \end{array}$$



### 5.2.2 CALCULATING THE ENDPOINTS

Rather than perform the complete matrix multiply, only those portions of the matrix that are required for a given situation will be calculated. The transformed  $x'$  &  $y'$  coordinates ( $x'$   $y'$ ) taken by themselves represent the orthographic projection. To change this to a true perspective value,  $x'$  &  $y'$  must each be divided by  $z'$  to become  $x''$  and  $y''$  (this is the approach that will be taken).

The formulae for  $x'$ ,  $y'$  and  $z'$  are:

$$x' = m(x \cos a + z \sin a)$$

$$y' = m(x \sin a \sin b + y \cos b - z \cos a \sin b)$$

$$z' = x \sin a \cos b - y \sin b - z \cos a \cos b$$

where  $m$  = magnification or scaling factor

For ease and speed of repetitive computation the above formulae can be re-written as:

$$x' = m(x \cos a + z \sin a)$$

$$y' = m(x Y1TERM + y \cos b - z Y2TERM)$$

$$z' = x Z1TERM - y \sin b - z Z2TERM$$

where:

$$Y1TERM = \sin a \sin b$$

$$Y2TERM = \cos a \sin b$$

$$Z1TERM = \sin a \cos b$$

$$Z2TERM = \cos a \cos b$$

### 5.2.3 SELECTING THE VISIBLE FACETS

The selection of which Facets are visible is determined by calculating the  $z'$  coordinate for all the vertex and normal points; if the  $z'$  of the normal point is less than the  $z'$  of the point associated with it on the Facet, then the Facet is visible.

### 5.2.4 CLIPPING THE LINE SEGMENTS TO WINDOW BOUNDARIES

### 5.2.5 DRAWING THE LINE SEGMENTS

### 5.3 DISPLAY LIST PLOTTING

### 5.4 MOVING THE DISPLAY BUFFER TO THE SCREEN BUFFER

### 6.0 FORTH SOURCE CODE

Cohen-Sutherland  
routine used

See p. 65

"PRINCIPLES OF  
INTERACTIVE  
COMPUTER  
GRAPHICS"

Note: for quick  
multiplication  
a word like  
clipping is rare.

$$z' = xA - yB - zC$$

Assembly language  
See June 83 Data Display  
Journal.





