INSTRUCTIONAL GRAPHICS PACKAGES TO BE USED WITH A LINE PRINTER

A.L. Furtado
A.A.B. Furtado
F.A. Messeder

Departamento de Informatica
Pontifícia Universidade Católica do Rio de Janeiro
Brasil

Introduction

It has been claimed that introductory courses on programming, with an emphasis on problem-solving, should not be restricted to purely numerical applications. In [1] (see also [5]) a number of elementary graphics problems are proposed, to be attacked by students using the UCSD Pascal implementation, assuming that the installation has the appropriate graphics equipment. Graphics are particularly relevant in the case of engineering students, being the basis for practical applications in the area of computer aided design.

Unfortunately not all installations have graphics hardware. Perhaps even more frequent is the case where the equipment is present but not in a scale sufficient to be made accessible to large classes of undergraduate students.

The solution adopted in our university was to append a sub-program library to the WATFIV-S [3] compiler to provide a rudimentary but hopefully adequate graphics capability, using a line printer. WATFIV-S is a structured dialect of FORTRAN (rather close to the 1977 standard) designed and supplied by the University of Waterloo. The compiler is core-resident and issues good compile - and run-time error messages. Another slightly more versatile package was produced in Pascal, to be used by our M.Sc. students in computer science, as a first exposure to graphics. In this paper we shall concentrate on the WATFIV-S package;details about both packages are contained in [2].

2. The WATFIV-S package

The sub-program headings and their usage are explained below. A number of "system" variables are shared through a COMMON area, which however does not have to be declared in the users'programs.

2.1. SUBROUTINE CLEAR(C)

Initializes the screen, filling it with the character supplied in paramenter C. In the screen both the x- and the y- coordinates go from -35 to 35. CLEAR places the "pen" at point (0,0), in the up position, turned along the zero-degree angle. Often, but not necessarily, the character in C will be a blank; any character can be used.

2.2. SUBROUTINE COLOR(C)

Defines as C the character to be employed in the next lines to be drawn as the pen moves. If C is zero the pen goes to the up position and nothing is drawn as it moves, whereas with any other character the pen is down. Note that, if the screen has been initialized with a non-blank character and COLOR is called with C being a blank, the drawing will be done in "negative".

2.3. SUBROUTINE TURN(ANGLE)

The positive or negative integer in ANGLE is added to the angle along which the pen is directed. The angle is kept in degrees.

2.4. SUBROUTINE TURNTO(ANGLE)

The positive, null or negative integer in ANGLE becomes the angle along which the pen is directed.

2.5. SUBROUTINE MOVE(DIST)

The pen moves, in the direction of the current angle, DIST screen units, where DIST is an integer. If the pen is up it is merely displaced; otherwise a straight line is drawn with the current character. If DIST is zero and the pen is down, the character is drawn at the current position but the pen is not displaced.

2.6. SUBROUTINE MOVETO (XPOS,YPOS)

The pen moves to the point whose coordinates are given in screen units by the integers (XPOS,YPOS). If the pen is up it is merely displaced; otherwise a straight line is drawn with the current character. If the coordinates of the current point already are (XPOS,YPOS) and the pen is down, the character is drawn at the current position but the pen is not displaced.

2.7. SUBROUTINE WHERE(XPOS,YPOS,DIR)

The coordinates of the current point and the current angle are assigned to the integer variables XPOS,YPOS,DIR, respectively.

2.8. SUBROUTINE SHOW

The contents of the screen are printed on the line printer.

2.9. INTEGER FUNCTION ROUND(X)

The value of the real X is rounded to the next integer. If X is positive, 0.5 is added to its value before truncation; if X is negative, -0.5 is added. This is an auxiliary function, called from some of the sub-routines.

Note: MOVE and MOVETO issue error messages if their execution would cause the pen to wander off the screen. The contents of the screen prior to the erroneous move are printed and the program execution is terminated.

```
    SUBROUTINE CLEAR(C)
      INTEGER SYSA,I,J
      REAL SYSX,SYSY
      CHARACTER C*1,SYSTAB*1(71,89),SYSC*1
      COMMON/SYS/SYSTAB,SYSA,SYSX,SYSY,SYSC
      DO 1 I=1,71,1
         DO 2 J=1,89,1
            SYSTAB(I,J)=C
2        CONTINUE
1     CONTINUE
      SYSX=45.0
      SYSY=36.0
      SYSA=0
      SYSC='0'
      RETURN
      END

    SUBROUTINE COLOR(C)
      REAL SYSX,SYSY
      INTEGER SYSA
      CHARACTER C*1,SYSC*1,SYSTAB*1(71,89)
      COMMON/SYS/SYSTAB,SYSA,SYSX,SYSY,SYSC
      SYSC = C
      RETURN
      END

    SUBROUTINE TURN(ANGLE)
      INTEGER ANGLE,SYSA
      REAL SYSX,SYSY
      CHARACTER SYSTAB*1(71,89),SYSC*1
      COMMON/SYS/SYSTAB,SYSA,SYSX,SYSY,SYSC
      SYSA = MOD(SYSA + ANGLE,360)
      RETURN
      END

    SUBROUTINE TURNTO(ANGLE)
      INTEGER ANGLE,SYSA
      REAL SYSX,SYSY
      CHARACTER SYSC*1,SYSTAB*1(71,89)
      COMMON/SYS/SYSTAB,SYSA,SYSX,SYSY,SYSC
      SYSA = MOD(ANGLE,360)
      RETURN
      END

    SUBROUTINE MOVE(DIST)
      INTEGER DIST,SYSA,IX,IY,NX,NY,K,L,ROUND
      REAL RADS,SYSX,SYSY,NEWX,NEWY,X,Y,DX,DY
      CHARACTER SYSTAB*1(71,89),SYSC*1
      COMMON/SYS/SYSTAB,SYSA,SYSX,SYSY,SYSC
      IF(DIST .NE. 0) THEN DO
        RADS = SYSA * 0.0174532925
        NEWX = DIST*1.25*COS(RADS) + SYSX
        NEWY = DIST*SIN(RADS) + SYSY
        NX = ROUND(NEWX)
        NY = ROUND(NEWY)
        IF(NX .GT. 89 .OR. NY .GT. 71  .OR.
*          NX .LT. 1 .OR. NY .LT. 1) THEN DO
          SYSTAB(ROUND(SYSY),ROUND(SYSX)) = '∂'
          CALL WHERE(IX,IY,K)
          SYSX = NEWX
          SYSY = NEWY
          CALL WHERE(NX,NY,K)
          PRINT, ' ERROR IN MOVE'
          PRINT, ' PEN IS NOW AT (',IX,IY,')ANGLE',K
          PRINT, ' PEN WOULD BE AT (',NX,NY,')'
          CALL SHOW
          STOP
        END IF
        IF(SYSC .NE. '0') THEN DO
          IX = ROUND(SYSX)
          IY = ROUND(SYSY)
          L = IABS(NX - IX)
          IF(IABS(NY - IY) .GT. L) THEN DO
            L = IABS(NY - IY)
          END IF
          DX = FLOAT(NX - IX)/FLOAT(L)
          DY = FLOAT(NY - IY)/FLOAT(L)
          X = IX + 0.5
          Y = IY + 0.5
          DO 1 K = 1,L,1
            SYSTAB(IFIX(Y),IFIX(X)) = SYSC
            X = X + DX
            Y = Y + DY
1         CONTINUE
          SYSTAB(NY,NX) = SYSC
        END IF
        SYSX = NEWX
        SYSY = NEWY
      ELSE DO
        IF (SYSC .NE. '0') THEN DO
          SYSTAB(ROUND(SYSY),ROUND(SYSX)) = SYSC
        END IF
      END IF
      RETURN
      END

    SUBROUTINE MOVETO(XPOS,YPOS)
      INTEGER XPOS,YPOS,SYSA,IX,IY,NX,NY,K,L,ROUND
      REAL SYSX,SYSY,NEWX,NEWY,X,Y,DX,DY
      CHARACTER SYSTAB*1(71,89),SYSC*1
      COMMON/SYS/SYSTAB,SYSA,SYSX,SYSY,SYSC
      IF(IABS(XPOS) .GT. 35 .OR. IABS(YPOS) .GT.
*       35) THEN DO
        SYSTAB(ROUND(SYSY),ROUND(SYSX)) = '∂'
        CALL WHERE(IX,IY,K)
        PRINT, ' ERROR IN MOVETO'
        PRINT, ' PEN IS NOW AT   (',IX,IY,')'
        PRINT, ' PEN WOULD BE AT (',XPOS,YPOS,')'
        CALL SHOW
        STOP
      END IF
      NEWX = XPOS * 1.25 + 45.0
      NEWY = YPOS + 36.0
      NX = ROUND(NEWX)
      NY = ROUND(NEWY)
      IX = ROUND(SYSX)
      IY = ROUND(SYSY)
      IF((NX .NE. IX) .OR. (NY .NE. IY)) THEN DO
        IF(SYSC .NE. '0') THEN DO
          L = IABS(NX - IX)
          IF(IABS(NY - IY) .GT. L) THEN DO
```

```
     L = IABS(NY - IY)
    END IF
    DX = FLOAT(NX - IX)/FLOAT(L)
    DY = FLOAT(NY - IY)/FLOAT(L)
    X = IX + 0.5
    Y = IY + 0.5
    DO 1 K = 1,L,1
     SYSTAB(IFIX(Y),IFIX(X)) = SYSC
     X = X + DX
     Y = Y + DY
1   CONTINUE
    SYSTAB(NY,NX) = SYSC
   END IF
   SYSX = NEWX
   SYSY = NEWY
  ELSE DO
   IF(SYSC .NE. '0') THEN DO
    SYSTAB(IY,IX) = SYSC
   END IF
  END IF
  RETURN
  END
```

```
 SUBROUTINE WHERE(XPOS,YPOS,DIR)
  INTEGER XPOS,YPOS,DIR,SYSA,ROUND
  REAL SYSX,SYSY
  CHARACTER SYSC*1,SYSTAB*1(71,89)
  COMMON/SYS/SYSTAB,SYSA,SYSX,SYSY,SYSC
  XPOS = ROUND((SYSX-45.0)*0.8)
  YPOS = ROUND(SYSY-36.0)
  DIR = SYSA
  RETURN
  END
```

```
 SUBROUTINE SHOW
  INTEGER I,J,SYSA,M
  REAL SYSX,SYSY
  CHARACTER SYSTAB*1(71,89),SYSC*1
  COMMON/SYS/SYSTAB,SYSA,SYSX,SYSY,SYSC
  PRINT 7
7 FORMAT('1')
  DO 6 I=1,71,1
   M = 72 - I
   PRINT 8,(SYSTAB(M,J),J=1,89,1)
8  FORMAT(' ',7X,89A1)
6 CONTINUE
  PRINT 7
  RETURN
  END
```

```
 INTEGER FUNCTION ROUND(X)
  REAL X
  IF(X .GT. 0.0) THEN DO
   ROUND = X + 0.5
  ELSE DO
   ROUND = X - 0.5
  END IF
  RETURN
  END
```

3. An Example

The sample program below draws a "spirolateral", an example taken from [1]. The input data are: angle=120, size=10, m=3, n=5, seq=rrℓℓr. The result illustrates the "negative" drawing option.

See figures at the end.

```
$JOB        FUR,KP=29,NOEXT,NOWARN
     CHARACTER*1 SEQ(10)
     INTEGER M,N,I,K,ANGLE,SIZE,X,Y,A
     READ, ANGLE,SIZE,M,N
     DO 1 I = 1,N
      READ10, SEQ(I)
10    FORMAT(A1)
1    CONTINUE
     CALL CLEAR('#')
     CALL MOVETO(0,17)
     CALL COLOR(' ')
     DO 2 K = 1,M
      DO 3 I = 1,N
       CALL MOVE(SIZE * I)
       IF(SEQ(I) .EQ. 'R') THEN DO
        CALL TURN(-ANGLE)
       ELSE DO
        CALL TURN(ANGLE)
       END IF
3     CONTINUE
2    CONTINUE
     CALL SHOW
     STOP
     END
```

4. Remarks

The position of the pen is kept internally in floating-point representation in order to avoid possible loss of precision, caused by successive calls to MOVE. Both MOVE and MOVETO are based on the simple DDA algorithm [4].

The scale factor used to balance the horizontal and vertical displacements was determined assuming that the printer works with 10 characters per inch horizontally and 8 characters per inch vertically. The actual "screen" dimensions are another machine-dependent consideration, being related to the size of the printed page.

In the Pascal package we provided the ability to produce contrast through grey-scale levels (indicated by an integer in the sub-range 0..7, passed as argument of procedure PENCOLOR), achieved by suitable overprinting patterns [6]. Also part of the contrast feature is a recursive procedure FILL [4], which paints a closed area in the indicated grey-scale level. A project to design a high-level language (like [7], for example) to be compiled or pre-processed into calls to the basic procedures is being contemplated.

With instructional packages, in general, motivation is a crucial aspect. The pictures obtained with a line printer cannot, of course, be compared with those produced with more specialized equipment. Figure 3.1 in reference [1] is a case in point: although we managed to draw it on our line printer the result was barely recognizable. Accordingly, one must restrict oneself to pictures that are not very crowded. Finally, the applications should be related to the subject area of each student class, with a more artistic or technological bent, according to the case.

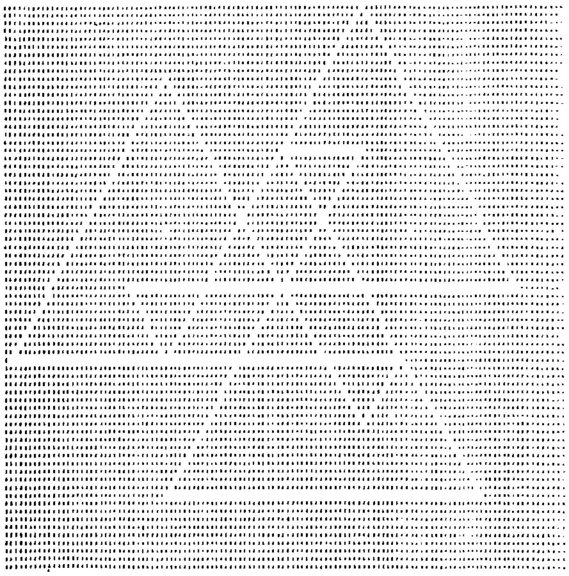
Acknowledgement

The authors are grateful to M.Stanton for help-

ful suggestions.

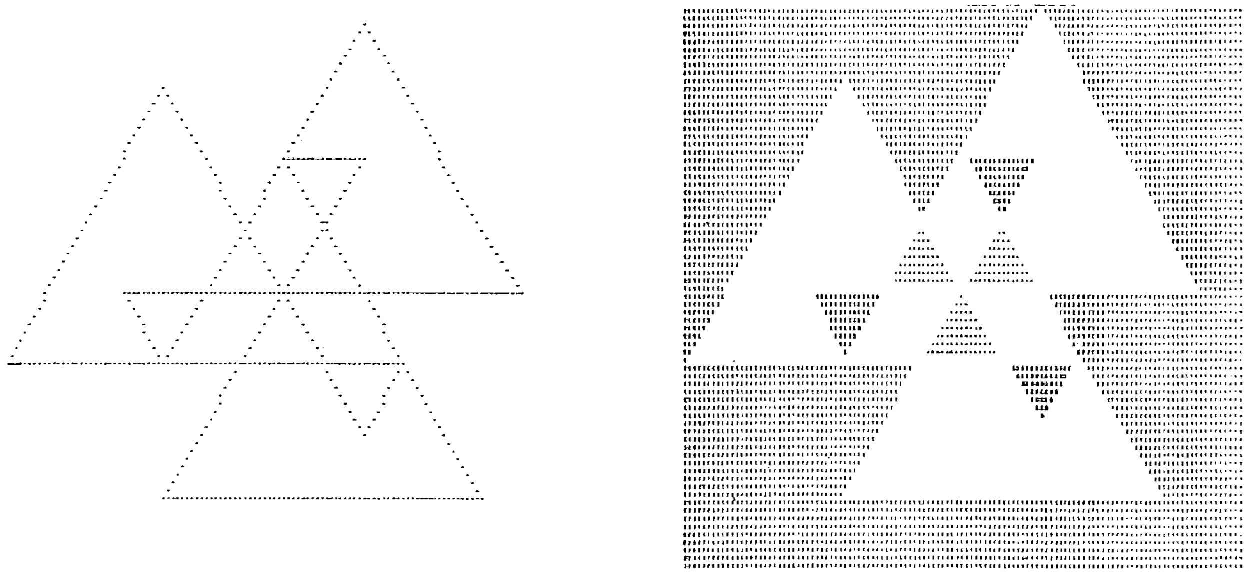
References

1. K.L. Bowles - "Problem solving using Pascal" - Springer-Verlag (1977).
2. A.L. Furtado, A.A.B. Furtado and F.A.Messeder- Instructional graphics packages to be used with a line printer - technical report, Pontifícia Universidade Católica do R.J. (1982).
3. R.C. Holt and J.N.P. Hume - "Fundamentals of structured programming using FORTRAN with SF/K and WATFIV-S" - Reston (1977).
4. W.M. Newman and R.F. Sproull - "Principles of interactive computer graphics" - McGraw-Hill, (1979).
5. S. Papert - "Uses of technology to enhance education" - AI-M-298, The Artificial Intelligence Lab., MIT (1973).
6. L. Press - "Computer and serial imagery" in "Artist and Computer" - R.Leavitt (ed.) - Creative Computing Press (1976).
7. C.J. van Wyk - "A high-level language for specifying pictures - ACM Trans. on Graphics- (April, 1982).



b. in "negative"

Figure: a spirolateral



a. in "positive"

c. in 4 grey-scale levels