2333 W. Huron
Chicago, Il 60612
October 23, 1980

Robert Fabris
3626 Morrie Drive
San Jose, CA 95127

Dear Bob,

I've written a bunch of BASIC programs to allow users to interactively construct line drawings with the joystick and to rotate, scale, and translate them once they've been drawn. The documentation is different in that I typed copies of all my programs on a minicomputer. This was extra work since any change in the BASIC code had to be updated on the mini, but it was the only way to get printouts easily.

The printout I've sent contains all the programs in line number order and the comments should be adequate to explain what's going on. Basically LDRAW (line 1500) lets you use joystick 1 to draw lines with mode 0 or 1. Each xy coordinate and drawing mode is packed into the @ array by PACK.

To scale a picture up (scaling down is a little trickier and you can't do both in the same program because of non-linearities) you use SETSCALE (2200), which reads the joystick and calls UNPK to unpack each stored xy coordinate. Then we call SCALE to perform the transformation.

Obviously all the programs don't need to be in memory at once and in fact they probably won't all fit. There's lots of BASIC comment lines that can be removed. Of course any change requires you to use LDRAW to refill the @ array, since its location changes.

Since rotation requires using sines and cosines, I store the cosines of 0->90 degrees scaled up by 250 (cosines are < 1 and BASIC doesn't handle fractions) using a program Dan Sandin got from someone (COSLD). So it's necessary to GOSUB 1000 to set up the cosine array whenever the BASIC code is modified. So the packed xy coordinates from LDRAW start at @(91). If you're not using the rotate code, the code can be changed to use all of the array elements for packed xy coordinates and COSLD and COSSIN can be removed.

The COSSIN program calculates the sine and cosine of any angle (positive or negative). We only need to store the 91 cosines since all others can be derived from these. If you're doing rotates, the variable R is used in a computed GOSUB (line 2385), so for example if you want to rotate about the x axis, set R=1200 (entry point to XROT). I've also got YROT (1300), ZROT (1400), and I derived the equations for x followed by y rotation in XYROT (2400). Note that y followed by x is not the same. Ambitious folks who enjoy multiplying matrices can derive other equations for xz, yz, xyz, xzy, etc. rotations.

In each rotation I give the formula for the new x and y coordinate (x' and y') in terms of the old x, y, and z. Since for now LDRAW works in 2D, z is always 0. In line 2410 and 2420 what I'm giving are the equations for the new x and y when we rotate about the x axis by the angle t (theta) and about the y axis by the angle g (gamma).

P.S. Here's the $12.50 for my renewal. Thanks.

Jim Marselle

```
**********************************
UNPK
        Jim Marselle    6/2/80

This program unpacks x, y, and p from the value in @(i).

needs:
        @(i).

returns:
        x, y, p.

clobbers:
        a.

calls:
        nothing.
**********************************
500 .unpk x , y, & p.
510 a = @(i)
520 p = 1
530 if a < 0 p = 0 ; a = -a
540 x = a / 100 - 80 ; y = rm - 50
550 return
**********************************
**********************************
COSSIN
        Jim Marselle    5/27/80

This program calculates the sin and cos of t in c and s, resp.

needs:
        t.

returns:
        c, s.

clobbers:
        u, v.

calls:
        nothing.
**********************************
700 .cos & sin of t in c
710 .& s. clobbers u & v.
720 u = t / 360 ; u = rm
730 v = abs(u)
740 .cos(-t) = cos(t)
750 if v < 91 c = @(v) ; goto 800
760 if v < 181 c = -@(180 - v) ; goto 800
770 if v < 271 c = -@(v - 180) ; goto 800
780 c = @(360 - v)
790 .sin(t) = cos(90 - t)
791 .sin(-t) = -cos(90 - t)
800 if v < 91 s = @(90 - v) ; goto 850
810 if v < 181 s = @(v - 90) ; goto 850
820 if v < 271 s = -@(270 - v) ; goto 850
830 s =-@(v - 270)
850 if u < 0 s = -s
860 return
**********************************
PACK
        Jim Marselle    5/27/80

This program packs x, y, and p into @(i).
```

needs:
        i, p, x, y.

returns:
        @(i) = packed value.

clobbers:
        nothing.

calls:
        nothing.

```
*****************************
900 .pack p , x , y into @(i)
910 .100 * (x + 80) + y + 50
920 if p = 1 @(i) = 100 * x + y + 8050 ; return
930 @(i) = -100 * x - y - 8050 ; return
*****************************
*****************************
```

COSLD

        Dan Sandin   5/27/80

This program loads @(0) -> @(90) with the cosines of 0 -> 90 degrees times 250 (to keep roundoff as small as possible).

needs:
        enough @() space.

returns:
        nothing.

clobbers:
        m, q, v, x.

calls:
        nothing.

```
*****************************
1000 .cos array load
1010 x = 10000
1020 m = 2715
1030 v = 0
1060 @(0) = 250
1070 for q = 1 to 90
1080 v = v - x / m
1090 x = x + v
1095 @(q) = x / 40 ; .* 250
1100 next q
1110 return
*****************************
*****************************
```

XROT

        Jim Marselle   5/27/80

This program calculates transformed coordinates for a rotation about the x axis at the angle whose cosine = c and sine = s.

needs:
        x, y, z = original pix coordinates. (note: z = 0 if we're 2D).
        c and s.

returns:
        u, v = new pix x-y coordinates.

clobbers:
        nothing.

calls:
nothing.
****
1200 .x rot, new x, y, z in
1210 .u, v, w, needs c & s
1220 .x' = x , y' = ycos + zsin , z' = zcos - ysin
1230 u = x
1240 v = (y * c + z * s) / 250
1250 return
****
****
YROT
Jim Marselle 5/27/80

This program calculates transformed coordinates for a rotation about the y axis at the angle whose cosine = c and sine = s.

needs:
x, y, z = original pix coordinates. (note: z = 0 if we're 2D).
c and s.

returns:
u, v = new pix x-y coordinates.

clobbers:
nothing.

calls:
nothing.
****
1300 .y rot
1310 .x' = xcos - zsin , y' = y , z' = xsin + zcos
1320 u = (x * c - z * s) / 250
1330 v = y
1340 return
****
****
ZROT
Jim Marselle 5/27/80

This program calculates transformed coordinates for a rotation about the z axis at the angle whose cosine = c and sine = s.

needs:
x, y, z = original pix coordinates. (note: z = 0 if we're 2D).
c and s.

returns:
u, v = new pix x-y coordinates.

clobbers:
nothing.

calls:
nothing.
****
1400 .z rot
1410 .x' = xcos + ysin , y' = ycos - xsin , z' = z
1420 u = (x * c + y * s) / 250
1430 v = (y * c - x * s) / 250
1440 return
****
****
LDRAW
Jim Marselle 6/2/80

```
This pgm allows the user to draw lines with mode 0 (no draw) and mode 1
(draw). Use jx(1) and jy(1) to control the rubber-band line. pull tr(1) to
draw the line, if kn(1) < 0, mode = 0. Otherwise mode = 1.
To delete the last drawn point hit a key in column 1, to exit hit a key
in column 2.

needs:
        Enough @() space to store the packed x, y, and p values starting at
        @(92) (the number of points will
        be stored at @(91)). This is done to allow @(0) -> @(90)
        to contain the cosines of 0 -> 90 degrees for rotations. If rotation
        will not be used, change this pgm.

returns:
        @() filled with the packed values of x, y, and p (the draw mode).
        b and c are the x and y center of the pix, resp.

clobbers:
        b, c, d, e, i, p, u, v, x, y.

calls:
        PACK, UNPK.
*********************************
1500 .line draw
1510 nt = 1
1560 clear
1570 cy = 40
1580 print "jx(1) , jy(1) to move"
1590 print "tr(1) to draw line"
1600 print "if kn(1) > 0 , we draw(p = 1)"
1610 print "else we don't (p = 0)"
1620 print "hit 1 to delp"
1630 print "2 to quit"
1640 i = 92 ; x = 0 ; y = 0 ; p = 0
1641 .xmax, xmin, ymax, ymin
1642 b = -80 ; c = 79 ; d = b ; e = c
1650 .pack dummy first point into @(92)
1660 gosub 900
1670 line 0 , 0 , 0
1680 input b ; clear
1685 u = 0 ; v = 0 ; i = 93
1690 .quit?
1700 if &(22) @(91) = i - 92 ; goto 1890
1710 u = u + jx(1) ; v = v + jy(1)
1720 line u , v , 1 ; line x , y , 0
1721 line u , v , 2 ; line x , y , 0
1740 if tr(1) goto 1820 ; .line
1750 if &(23) = 0 goto 1700
1760 .delp
1770 if i = 93 goto 1700
1780 i = i - 2
1790 gosub 500 ; .unpk x , y , p
1791 line x , y , 0
1792 u = x ; v = y
1793 i = i + 1
1795 gosub 500
1800 if p line x , y , 2 ; line u , v , 0
1805 x = u ; y = v
1810 if &(23) mu = 100 ; goto 1810
1812 goto 1700
1820 if kn(1) > 0 p = 1 ; goto 1840
1830 p = 0
1840 x = u ; y = v
1850 line x , y , p
1860 gosub 900
1861 if x > b b = x ; goto 1863
```

```
1862 if x < c c = x
1863 if y > d d = y ; goto 1870
1864 if y < e e = y
1870 i = i + 1
1871 if tr(1) mu = 100 ; goto 1871
1880 goto 1700
1890 clear
1891 .get xcen & ycen
1892 b = (b + c) / 2
1893 c = (d + e) / 2
1900 for i = 92 to @(91) + 91
1910 gosub 500
1920 line x , y , p
1930 next i
1940 return
**********************************************
**********************************************
```

MOV
Jim Marselle 6/2/80

This program calculates the transformed x-y coordinates for a translation of k units in x, l units in y, and returns the transformed coordinates in u and v, resp.

needs:
k, l, x, and y.

returns:
u and v = new x-y coordinates.

clobbers:
nothing.

calls:
nothing.

```
*********************************
2000 .mov
2010 .x' = x + tx
2020 .y' = y + ty
2030 u = x + k
2040 v = y + l
2050 return
*****************************************
*****************************************************
```

SCALE
Jim Marselle 6/2/80

This program calculates the transformed x-y coordinates for a scale (up) of factor k in x and l in y. We return the transformed coordinates in u and v, resp. Since we assume the pix is not centered at the origin we actually perform a concatenation of the transformations translate-to-origin, scale, and translate-back. To do this we need the x center and y center of the pix in b and c, resp.

needs:
b, c, k, and l.

returns:
u and v, the new x-y coordinates.

clobbers:
nothing.

calls:
nothing.

```
*****************************
```

```
2100 .scale
2110 .x' = xSx + Tx(Sx - 1)
2120 .y' = ySy + Ty(Sy - 1)
2130 u = (x - b) * k + b
2140 v = (y - c) * l + c
2150 return
```

****

****

SETSCALE

Jim Marselle 6/2/80

This program allows the user to scale up a pix in the x and y dimensions by using kn(1) (We could use kn(2) for y scaling). When the user pulls tr(1) another copy of the pix is drawn. Hit a key in column one to exit.

needs:

@() set up.

returns:

nothing.

clobbers:

i, k, l, p, u, v, x, y.

calls:

UNPK, SCALE.

****

```
2200 .set scale
2205 if &(23) return
2210 if tr(1) = 0 goto 2205
2215 if tr(1) mu = 100 ; goto 2215
2220 . k = kn(1) + 129 ; .down
2230 k = kn(1) / 40 + 4 ; .up
2235 l = k
2240 for i = 92 to @(91) + 91
2245 gosub 500
2250 gosub 2100
2255 line u , v , p
2260 next i
2270 goto 2205
```

****

****

ROT

Jim Marselle 6/2/80

This program rotates a pix about the axis specified by r (r is the entry point of the desired coordinate transformation routine, e.g. XROT, YROT, etc.) We use kn(1), kn(2), and kn(3) to specify the rotation angle about the x, y, and z axes, resp. These angles are saved in t (theta), g (gamma), and p (phi). The sines of t, g, and p are s, f, and h, resp.
The cosines of t, g, and p are c, d, and e, resp.
To draw a rotated pix, pull tr(1), to exit hit any key in column 1.

needs:

@() set up and r, the entry point of the desired rotation routine.

returns:

nothing.

clobbers:

c, d, e, f, h, i, p, t.

calls:

COSSIN, UNPK, and the ROT routine specified by r.

****

```
2300 .rot
2310 if &(23) return
2320 if tr(1) = 0 goto 2310
2330 if tr(1) mu = 100 ; goto 2330
2340 t = kn(1) * 180 / 128
2341 g = kn(2) * 180 / 128
2342 p = kn(3) * 180 / 128
2343 a = t ; t = g ; gosub 700
2344 d = c ; f = s
2345 t = p ; gosub 700
2346 e = c ; h = s
2347 t = a ; gosub 700
2360 for i = 92 to @(91) + 91
2380 gosub 500
2385 gosub r
2390 line u , v , p
2395 next i
2396 goto 2310
**********
**********
```

XYROT

Jim Marselle 6/2/80

This program calculates transformed coordinates for a rotation about the x followed by y axes at the angles whose cosines = c and d, resp. and whose sines = s and f, resp.

needs:

x, y, z = original pix coordinates. (note: z = 0 if we're 2D).
c, s, d, f.

returns:

u, v = new pix x-y coordinates.

clobbers:

nothing.

calls:

nothing.

```
**********
2400 .xyrot
2410 .x' = xcosg + ysint sing - zsing cost
2420 .y' = ycost + zsint
2430 . cos t, g, p = c, d, e.  sin t, g, p = s, f, h.
2440 u = x * d / 250
2441 u = u + (y * s / 250 * f / 250)
2442 u = u - (z * f / 250 * c / 250)
2450 v = y * c / 250 + z * s / 250
2460 return
**********
**********
```

SETMOV

Jim Marselle 6/2/80

This program allows the user to translate a pix in the x and y directions by using jx(1) and jy(1), resp. We draw a flashing box which indicates the center of where the pix will be drawn.
When the user pulls tr(1) another copy of the pix is drawn. Hit a key in column one to exit.

needs:

@() set up, b and c = x center and y center, resp.

returns:

nothing.

```
clobbers:
        i, k, l, m, n, p, u, v, x, y.

calls:
        UNPK, MOV.
************************
2500 .setmov
2505 m = 0 ; n = 0
2510 if &(23) return
2515 m = m + jx(1) ; n = n + jy(1)
2520 box m , n , 2 , 2 , 1
2525 box m , n , 2 , 2 , 3
2530 if tr(1) = 0 goto 2510
2535 if tr(1) mu = 100 ; goto 2535
2540 k = m - b ; l = n - c
2545 for i = 92 to @(91) + 91
2550 gosub 500
2560 gosub 2000
2570 line u , v , p
2580 next i
2590 goto 2510
************************
```