BASIC POSTSCRIPT

The POSTSCRIPT commnands listed below are only a small subset of the language. But if you acquire a working knowledge of what they can do for you they will be sufficient for all your needs in this course. This list should be used only as a guide in conjuction with the sample programs distributed in class and the postscript programs that we will make available in the public directory. For a deeper understanding of this primitive but powerful graphics language you should consult a POSTSCRIPT manual. The Addison-Wesley publication

PostScript Language Tutorial and Cookbook

by "Adobe Systems Incorporated" (1985) is one of many references you can consult for this purpose.

In PostScript all input data or user inserted data are initially successively placed into and used from a basic stack. A stack is a Data Structure where successive items are stuffed in on top of each other. The only way to access any item in the stack is to move that item on the very top of the stack. This can be done by removing or using one by one all the elements above it, or by means of successive steps which carry out some of the operations listed below.

Stack Commands

pop

Discard the top element.

exch

Exchange the top two.

dup

Duplicate the top element.

copy

An integer n placed before **copy** will cause the duplication of the top n elements of the stack. That is the sequence of these elements will be found repeated twice in the stack.

Algebraic or transcendental operations use up data stored in the stack. Whether such operations use only the top element or several successive top elements of the stack the result is a permanent loss of these elements unless they are previously duplicated and/or stored.

Algebraic Commands

sub and **add**

The writing **x y sub** causes the placing of **x** then **y** on the stack (**y** on top of **x**) then the removal of both of them and the placing of $\mathbf{x} - \mathbf{y}$ on top of the stack. Likewise if **x** and **y** are already in the stack in that order, a simple invocation of **add** causes their removal and placement of $\mathbf{x} + \mathbf{y}$ on top of the stack. The following commands have analogous consequences.

mul

x y mul places the product **xy** on top of the stack.

div

x y div places the ratio **x/y** on top of the stack.

mod

a b mod places on top of the stack the remainder **r** of the integer division $\mathbf{a} = \mathbf{qb} + \mathbf{r}$ $(0 \leq r < b)$.

neg

x neg places $-\mathbf{x}$ on top of the stack.

abs

x abs places $|\mathbf{x}|$ (the absolute value of x) on top of the stack.

ceiling

"**x ceiling**" places on top of the stack the smallest integer greater than or equal to **x**.

floor

"**x floor**" places on top of the stack the greatest integer smaller than or equal to **x**.

round

"**x round**" places on top of the stack the closest integer to **x**. (Probably using the smaller if two integers are equidistant from **x**.)

sqrt

"**x sqrt**" places the square root of **x** on top of the stack.

rand

"**rand**" places a random integer in the range 0 to $2^{31} - 1 = 2147483647$ on top of the stack.

srand

"**x srand**" sets **x** to be the new "seed" for the built-in random number generator.

Transcendental Commands

The trascendental functions Arctangent, Cosine, Sine, Exponential, Natural Logarithm, base 10 Logarithm, are respectively invoked by the commands:

atan cos sin exp ln log

Graphics Commands

A number of graphic parameters are assumed by default at the start of the execution of a PostScript program. In particular the origin of the coordinate system is at the bottom of an $8.5in$ by $11in$ page. The x-axis is along the bottom border the page and the y-axis is the left border. The default scale is such that the points $(20,20)$, $(580,20)$, $(580,770)$, $(20,770)$ are respectively and aproximately at the SW, SE, NE and NW corners of the page. To check this just execute a file with the following sequence of postcript commands:

```
/mf
/Helvetica findfont 10 scalefont def
/mfb
/Helvetica findfont 25 scalefont def
/mfbb
/Helvetica findfont 35 scalefont def
/mfs
/Helvetica findfont 8 scalefont def
mfs setfont
20 20 moveto (SW) show
580 20 moveto (SE) show
580 770 moveto (NE) show
20 770 moveto (NW) show
20 20 moveto
580 20 lineto
580 770 lineto
20 770 lineto
closepath
5 setlinewidth stroke
showpage
```

The graphics parameters can be modified at will by the invocation of commands such as

"**x y moveto**"

this moves the current point to (x, y).

"**x y lineto**"

this draws a line from the current point to (x, y) and makes the latter the new current point.

"**a b rmoveto**"

if the current point is (x, y) this moves the current point to $(x + a, y + b)$.

"**a b rlineto**"

if the current point is (x, y) this draws a line from (x, y) to $(x + a, y + b)$ and makes the latter the new current point.

"**x setlinewidth**"

sets the thickness of lines to **x/72** of an inch.

"**stroke**"

causes the lines to be drawn

Finally, "**showpage**" is the command that causes the page to be printed

Note also that the "scale" can be changed by the command "**p q scale**" which causes the scale on the x-axis to change by the factor p and the scale on the y-axis to change by the factor q. The command "$\mathbf{x_o}$ $\mathbf{y_o}$ **translate**" causes the origin of the new coordinate system to be placed at the point whose coordinates in the current coordinate system are (x_o, y_o). The command α **rotate** causes the coordinate system to rotate counterclockiwise by the angle α (which should be given in degrees). Successive uses of **scale** and **translate** and **rotate** may cause nothing to appear on the page. The reason is that you inadvertently may have asked your laser printer to draw your desired picture just about on the moon! To remedy this problem we use the commands **gsave** and **grestore**. Before you do any wild motions make an invocation of **gsave**, this done carry out your motions. This done make an invocation of **grestore**, and this will will bring you back to earth or in any case wherever you were before you made your moves.

Loops and Procedures

The handouts should give you an idea on how PostScript procedures are constructed. The commands that cause the execution of a given procedure a given number of times are

for and **repeat**

The invocation of **for** in the form "**a s b pluto for**" causes the execution of **pluto** with an (invisible) an input parameter which takes the values $a, a + s, a + 2s, a + 3s, \ldots$ up to the last integer n such that $a + sn \leq b$. So **3 2 7 pluto for** has the same effect as if you wrote

3 pluto, **5 pluto**, **7 pluto**

repeat

The invocation of **m {pluto} repeat** causes the procedure **pluto** to be executed m times over.

NOTE If your procedure **pluto** requires the printing of the invisible loop parameter we called k or some specified function of it. The quantity to be printed must first be converted to a string. For instance if the quantity to be printed is **x**, and you predict it will require at most say, 5 characters to print, then write **cvs x** $(*****)$ before invoking **show**

Conditional executions

if

You may also cause the conditional execution of **pluto** by a command of the form "**bool pluto if** ". This causes the execution of **pluto** if the boolean expression **bool** evaluates to *true*. To construct boolean expressions you may use the commands below

eq " = " **ne** " ≠ " **ge** " ≥ " **gt** " > " **le** " ≤ " **lt** " < "

Each of them operates on the top two elements on the stack and replaces them with the result of the corresponding comparison. For instance **x y ge** places on the stack **true** if $x \geq y$ and **false** if $x < y$.

Miscelanea Graphics

You may find useful the commands

arc arcto curveto

their action is as follows

arc

The invocation **x y r alpha beta arc** causes drawing of the the portion of the circle with center at (x, y) and radius r that is delimited by the angles **alpha** and **beta** (given in degrees).

arcto

The invocation $\mathbf{x_1}$ $\mathbf{y_1}$ $\mathbf{x_2}$ $\mathbf{y_2}$ **r arc** causes drawing of the the portion of the circle with radius r that starts tangently to the line joining the current point to the point (x_1, y_1) and ends tangently to the line that joins (x_1, y_1) to (x_2, y_2). This command may be used to round sharp corners, in the drawing of successive lines.

curveto

Experiment with this command that should be invoked in the form " $\mathbf{x_1}$ $\mathbf{y_1}$ $\mathbf{x_2}$ $\mathbf{y_2}$ $\mathbf{x_3}$ $\mathbf{y_3}$ **curveto** ". It should append to your path the **Bezier** curve with control points (x_o, y_o), (x_1, y_1), (x_2, y_2), (x_1, y_1), (x_3, y_3), where (x_o, y_o) is the current point.

Note after you construct a path by a sequence of **lineto**'s followed by the command "**closepath**" you may fill the enclosed area with your desired color by the command:

h s b sethsbcolor

where **h** is the Hue value and **s** , **b** give you the "saturation" and the "brightness" respectively. You may set the latter two equal to 1 or experiment with them to see what effects you can achieve by changing them.