Appendix A

G. Wyvill 'Pictorial Description Language'

This computer graphics system provides:

Extension to an existing programming language

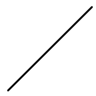
A suit of specialised programs for a particular job

A number of utility programs designed to operate on a common database

Examples of PDL

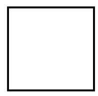
Example 1:

*DEFINE LINE *LINE 0,0 1,1 *PLOT LINE



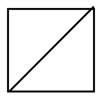
Example 2:

*DEF SQUARE *LI 0,0 1,1 1,0 0,0 *PL SQUARE



Example 3:

*DEF OBJ1 *TRANSFORM LINE *TRANSFORM SQUARE *PL OBJ1

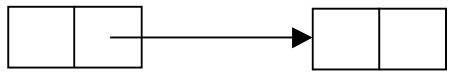


B. Kernighan 'PIC – A Language for Typesetting Graphics'

A picture in Pic is written as a sequence of statements. Statements can be used to specify a primitive object such as circle, ellipse, arc, spline or line. These objects may be a set of attributes such as position, text, height, width and line style, which may appear in any order. Positions may be described using absolute Cartesian coordinates and absolute sizes. These are sufficient, in that anything can be expressed in such terms, but they make it hard both to create the first draft of a picture and to modify it subsequently. Absolute values also discard any structural relationships that might exist among the objects in a picture. Pic provides two methods to help the user. One is the ability to link one object to another through positional and dimensional relationships. The other is the use of defaults. Objects sizes and positions and the relationships among objects all have sensible default values, so that simple pictures can be made with a minimal amount of specification.

Example1:

```
B:   box wid 0.5I ht 0.5I
     box same with .w at B.e
A:   arrow from last box.c right 1i
     box same with .w at A.end
     box same with .w at last box.e
```



Example2:

Each object class is defined in two parts: a declarative section and an instruction section. The declaration section contains variable declarations and relationships or constraints that must hold among the points of the object. This gives a system of simultaneous equations. The instruction section contains instructions for connecting points and for drawing other objects by invoking or calling them. A box can be defined with the procedure:

```
rect{
        var ne, nw, se, sw, wd, ht;
        nw = sw +(0,1)*ht;
        ne = nw +wd;
        se = sw + wd;
        conn ne to nw to sw to se to ne;
}
```

To draw an instance rect:

```
put rect{
        ht = 2;
        wd = 1;
        sw = 0;
}
```

There are actually many ways of drawing a rectangle, e.g. by giving one corner, one dimension and a relationship on the dimensions, or by giving three corners, or by giving two adjacent corners and the perpendicular dimension. Any method can be used provided enough information is given for the system to solve the equations.

```
put first: rect{
        sw = 0;
        ht = wd =1;
};
put second: rect{
        nw = first.se;
        ht = wd = first.ht;
};
put last: rect{
        sw = first.ne;
        se = next.ne;
        ht = wd;
};
```