SOFTWARE—PRACTICE AND EXPERIENCE, VOL. 4, 299–304 (1974)

Short Communications

Graphics under GEORGE 4

Like W. J. Milne [A Simple Interactive Graphics Facility, **4**, No. 1, 71–78 (1974)], we have also been developing cheap graphics on Tektronix 4010's connected to an ICL 1900, but have proceeded in a different way. The Atlas Computer Laboratory has a 1906A, running under GEORGE 4 with MOP (ICL's terminal control system), using an ICL 7903 as multiplexor. As we are running a service with a very uncooperative workload (in the sense that jobs do not fit together nicely), we do not normally allow interactive use, apart from editing. We do, however, have a lot of graphics users producing plotter and microfilm output, who need to see their results remotely.

Only two alterations have been made to the normal GEORGE MOP facilities. These allow the output of all, and the input of most, of the 128 possible even parity ASCII characters. Normally, a MOP terminal is restricted to the 64 character 'GRAPHIC' set, although a 3-shift character 'ALLCHAR' set exists. The first change to GEORGE allows ALLCHAR files to be created from a terminal containing the extra characters. The second change causes files marked as ALLCHAR to be treated as 3-shift, and hence send the correct characters to the terminal when listed. Thus graphics can be input from a terminal using INPUT and output to a terminal using LISTFILE. No user program is involved. So far, the input facility is used mainly to create documents containing upper and lower case. Clearly, the normal MOP facilities are available in full from such terminals.

Having no user code 'attached' to or interacting with the terminal makes the system very cheap to run. However, on-line use of the Tektronix is possible (though not encouraged). One method is to use 'conceptual multiplexors'—a GEORGE facility that allows programs to write directly to terminals, by-passing MOP. For reasons not fully understood, the response using this system is very poor. Also, the terminal loses its MOP capability. Another method is to use the ONLINE command to connect the terminal to the program as a pseudo tape reader, which will correctly deal with 3-shift input. Output is not so easy since it is necessary to output to an ALLCHAR file which is listed as it is produced. Control of the cross-hairs can be done by the user program.

It is encouraging to see more flexibility in the use of MOP terminals on 1900's, and it is hoped that ICL will take note of these requirements and developments.

P. E. Bryant
R. E. Thomas
Atlas Computer Laboratory
Chilton, Didcot, Berks

APL versus operator precedence

In an earlier letter [*Software—Practice and Experience*, **1**, No. 4, 411 (1971)], Professor Samet comments on APL's lack of operator precedence and right-to-left evaluation as an 'undesirable feature'. I would argue that the presence of operator precedence is particularly undesirable in a language such as APL.

Firstly, Professor Samet invokes the argument of 'common convention' for precedence and associativity relations. I am left wondering just what that common convention might be, for among FORTRAN, ALGOL 60, PL/1, PASCAL and SNOBOL4, *no two languages* give precisely the same precedence and associativity to that set of operators which the two languages have in common.

Secondly, 'common convention' (such as it is) extends only over a small number of simple arithmetic operations. Suppose for a moment that one were given the task of assigning precedences to APL operators—would left rotation be placed above or below set membership ? My own observations indicate that experienced ALGOL programmers are unable to state correctly the precedences of all of ALGOL 60's seventeen operators. (If you find an ALGOL programmer who can tell you which of $\equiv$ and $\supset$ has the higher precedence, chances are you have found an implementor.) A cursory glance at APL revealed at least thirty