the fourth program example. It is under a page long and finds all the sets of positions where you can put eight queens on a chess board so that no queen can capture another. The problem is solved recursively and if you want to see a computer work! Wow! It's really impressive and quite instructive if you want to understand recursion.

For small programs, a single keypress makes your entire program "visible". The manual shows you how to get the most out of the visible options for large programs. Quite a lot is available. You can look at the smallest detail in a procedure. When the procedure is thoroughly debugged and understood you can make it invisible. Go on to the next procedure and work on it. Check each and the interactions between the procedures. These tools really speed up the process of getting a program running properly.

Some of the other pluses for the system:

1. The controls you have over the program are many, and easy to learn and use. Example - if you don't like the way keys are assigned in the editor you can change them to be the same as your favorite editor.

2. It's great as a learning and exploring tool.

3. You can put in breakpoints for program testing.

4. The screen editor has templates available at a keystroke for the major Pascal structures. If you want a FOR template you just press the appropriate key and it appears at the cursor with the correct syntax. You just add the correct variables or numbers in the slots provided.

5. Since your program is saved as a Standard ASCII file, you can use it with other compilers after it has been written and debugged in Dr. Pascal.

Some of the minuses, all rather minor:

1. When the compiler reformats the program it removes white spaces, i.e., extra spaces unneeded by the compiler. For instance, if you write A = 3 + 7 - 55, it will give you A = 3+7-55. Personally, I find the extra spaces easier to read, and I make fewer mistakes when I write this way.

2. When you have run errors it is a little awkward correcting them. You will probably have to edit your program and start it running from the beginning.

Dr. Pascal works on all IBM PC compatibles with at least 512K, and most other MS - DOS and CP/M-86 systems. The developer is working on other systems such as the DEC Rainbow((. By the time you read this, Dr. Pascal may be ready for other systems. If you are interested, check with the company about availability and considerable discounts for quantity purchases by educational institutions.

Dr. Pascal is an excellent set of tools and a very good buy.

An Animation of Distillation
Part I
The Flame Vic Bendall, College of Natural and Mathematical Sciences, Eastern Kentucky University, Richmond, Kentucky 40475

The pedagogical value and overall attractiveness of computer aided instructional software is greatly enhanced by the incorporation of animation. Unfortunately good animation is much more time consuming to author than the routine program. To get fast animation, particularly when two or more actions appear to take place simultaneously on the screen, requires careful planning and logistical skills. In this series of articles, I will describe how the animation of a distillation was approached.

In the usual distillation most of the apparatus is static and can be easily drawn using conventional graphic commands on the monitor screen. The animated portion consists of a flickering flame to illustrate heating, a rotating stirrer bar in the heated liquid, liquid drops falling from the condenser and liquid accumulating in a collection vessel. Our task is to write a routine which will appear to execute all four actions at once. For illustrative purposes, we will write separate routines to perform each action separately and then combine them into a smoothly working whole.

This first article will describe the flickering flame routine. We need a routine which can be easily correlated with the subsequent routines. A BASIC program which shows a flame is given in the documentation of CHEMUTIL-2 (1). That routine works by defining nine 7x8 bit characters which each show a different view of a flame and having those characters available in the alternative character set. The nine characters are then printed successively directly on top of one another and a flickering flame results. The CHEMUTIL-2 documentation code cannot be used directly for the distillation because it is not general enough to allow the incorporation of other routines without excessive jerkiness in the combined routines. For example, rapid showing of the nine flame images followed by a falling drop and then nine more flames, will result in the flame being stopped while the drop falls. We need the drop to fall smoothly without the flicker being interrupted.

As a general rule, code written for a single application will be more efficient than that which can be easily adapted for other purposes. But general purpose code is more useful. Thus, we can write distillation code which is specific for that alone or the code can be written so that with just minor modifications, it would be used to demonstrate heating under reflux with or without a Dean-Stark trap. To allow for those and other future unanticipated uses, the code we shall write will be made as flexible as possible.

The first step is to allow for movement of the flame image to any screen location. In a distillation the flame is in a different relative posi-

tion with respect to the falling liquid than in a refluxing situation. Being able to move the flame independent of the falling drop and stirrer will made the coded routine more useful. It will certainly help when we attempt to combine flame and stirrer together. The code will be a loop of the form: -

Go to flame location: Draw flame image: Go to stirrer location: Draw stirrer image: Loop back

The stand alone flickering flame routine consistent with the previous loop will have to be in the form: -

Go to flame location: Draw a flame image: Loop back

The delay is deliberately introduced so that the routine can be tested independent of the stirrer and drop routines to be added later. Those routines will later replace the delay, but it is needed now so that we can be sure that the independent routine will be easily integrated with the others.

Now we cannot draw the same image of a flame each time through the loop or it will not flicker. We could blank out the flame image on alternate passes to make it flicker, but the result would be rather dull. The CHEMUTIL-2 method is to overprint nine different flame images rapidly. The loop will not be of the form: -

Go to flame location: Draw flame (1): Delay: Go to flame location: Draw flame (2): Delay: etc.

After nine flame images have been drawn, the loop starts over again.

Finally, we do not want the loops to be endless. There must be some way to turn off the flame. It could be done by stopping the execution of the code after a predetermined number of times through the loop. More generally useful is to us a keystroke to terminate it. The advantage of that is that the code now allows easy introduction of alternatives depending upon which key is struck. For example, striking the S-key could stop the animation and turn off the flame, while striking the C-key could leave the flame on but start the stirrer, too. The point is not that we intend to use the C-key in that way, but that we allow for that possibility before starting to write code. In a complex animation it is not easy to add additional features after coding without undesirable effects appearing.

A BASIC program to show a flickering flame can now be written. The following one is for Apple II+, IIe or IIc microcomputers and will show the flame on the text screen.

```
10 Home
20 For I = 97 to 105
30 VTAB 10 : HTAB 10
40 PRINT CHR$(I)
50 X - Peek(-16384):POKE(-16368),0: IF X=211 THEN I = 105:GOT0 80
60 NEXT I
70 GOTO 20
80 VTAB 10 : HTAB 10 : PRINT " "
90 VTAB 20 : END
```

The flame image is made up of successive images from ASCII(33) through ASCII(41). Line 50 stops the illusion when the S-key is struck and Line 80 extinguishes the flame. If CHEMUTIL-2 is available, BLOAD it and replace line 10 by CALL 25042:PRINT "&". The routine will show a true flame on the graphics screen.

BASIC is not the best language for this routine. A satisfactory illusion is obtained here only because the routine has no other tasks. The only delays are the usual ones as the computer reads and interprets each line. If even a small delay loop is added to simulate an additional task, the illusion is lost. For example, add the line: -

```
55 FOR K = 1 to 30 : NEXT K
```

Upon execution, the routine shows each character clearly and is unacceptable. Yet this delay is too short to allow the introduction of the stirrer and falling drop loops. We must resort to 6502 machine language code.

The source code shown was generated using the Toolkit Assembler (2) and used CHEMUTIL-2 to print the flame images contained in character set 2. To see this program in action you will need to enter the monitor (CALL-151) and enter the program directly. e.g. 7C10 : A9 OB 8D 00 7C A9 09 etc. Exit the monitor and BSAVE FLAME, A$7C00, L$8E then, with CHEMUTIL-2 in place, run the following BASIC program.

```
10 CALL 25042 : CALL 31760 : END
```

In the next part of this series, I will discuss and show a listing of a routine which shows a rotating stirrer bar suitable for combining with this flame sequence. Subsequent parts of this series will all require CHEMUTIL-2 for the successful execution of the code.

(1) Bendall, V. "CHEMUTIL-2, A chemistry Programming Utility"; Project SERAPHIM, NSFScience Education; Department of Chemistry, Eastern Michigan University, Ypsilanti, MI 48197, 1985.

(2) "Applesoft Tool Kit"; Apple Computer, Inc., 20525 Mariani Avenue, Cupertino, CA 95014.

BOOK REVIEW COLUMN

This issue's column includes books that cover a variety of computer topics, ranging from chip architecture to campus wide networking, and most readers should find at least one book that is of interest. As usual readers are invited (and urged) to express their opinions about this column. Whether you would like to review a book, suggest a topic area that doesn't receive enough coverage, or just make a general comment, write to Dr. Harry Pence, Professor of Chemistry, SUNY-Oneonta, Oneonta, NY 13820.

CAMPUS NETWORKING