Victor Bendall
Eastern Kentucky University

An Animation of Distillation
Part IV
The Falling Drop

This article includes the complete source code for the animated portion of a distillation graphic for an Apple II microcomputer. A BASIC listing is also included that draws the static surrounding graphic and CALLS the machine code that animates the flame, stirrer and falling drops described in the previous articles. There are some additional considerations when combining the operations that may not be immediately obvious from the code alone.

This routine does not use an internal clock directly to control the overall speed of the animation because there is no clock available in an unmodified Apple II microcomputer. The timing of the total animation is governed by how long it takes each individual animation loop to be executed. We can choose to flicker the flame x times for each drop falling or to have x drops falli for each flicker of the flame. In practice it is easier to sum several short intervals to make one long one than to divide the long interval into several short ones. This means that if the execution time for one flicker is shorter than that for a falling drop then it is easier to code , for example, seven flickers per drop than to code one seventh drop per flicker. The general rule for a complex animation is to use the loop with the shortest execution time as a standard. In our case, we can control the execution time for flicker or drop by using the delay loops but it seems more desirable to have fast flickering than high speed falling drops so that our code reflects the x flickers per drop option.

This same option has another advantage. It becomes very simple to not only show no drops but also to omit the drop loop completely. The x drops per flicker option makes it easy to show drops and no flicker but not the reverse. While we cannot be certain how this animation code may be used in a future application, it seems reasonable that we are more likely to want to retain the flame and omit the drops then the reverse. In the final code we use the flicker loop as the standard and the stirrer and drop routines are subsidary it.

Once a drop begins to fall the speed that it does so depends upon the execution time of the appropriate loop but that in turn depends upon the speed of the flame and stirrer loops. The falling drop, flame and stirrer loops are controlled by delay loops so that there is ample room for controlling the drop speed. However, we also need to be able to control how often a drop falls, There will be times when we want the liquid to distill slowly, or perhaps to reflux rapidly, and now is the time to code for possibilities. This combined code uses a new variable FREDRP to take care of them. By adjusting FREDRP and the old variable DRPCNT it is now possible to have fine control on the apparent rate of accumulation of ditillate.

The DRPIT code in this listing has been reorganized and should be compared with the DROPIT routine given in Part IV. The first few lines of the old code erased the last drawn drop of liquid displayed by the same loop the previous time it was executed. In that code only a delay lop was executed between successive executions of the drop loop. The result was satisfactory provided a new drop began to fall immediately after the old one splashed in the distillate. In the new code, the FREDRP variable was introduced so that there is a deliberately long interval between drops. The old code would now shhow a drop falling, then jumping back to the top position and hanging there. The reorganized code erases the last drawn drop before leaving the DROPIT loop and the undesirable effect is eliminated. This illustrates another point. In itself the old DROPIT code was satisfactory. It was only when it was combined with other individually satisfactory animation routines that a flaw showed up. However, our code was well

planned initially so that the change made to correct the flaw was relatively simple.

DROPIT has also been modified so that DRPCNT is used more rationally. In the earlier version DRPCNT was incremented each

time the DROPIT loop was executed. This meant that it did not count the number of drops directly. In the new code, DRPCNT is incremented each time a SPLASH is drawn. This gives a more rational control of the increase of the liquid pool since DRPCNT is now independent of the height through which the drop falls.

The source code for the complete animation should be entered through the Apple monitor starting at $ 7C15. The BASIC program uses that code and CHEMUTIL-2 (1) to demonstrate the complete distillation in action. Your copy of CHEMUTIL-2 should have the extra characters added as described in the earlier parts of this series. If those characters have not been added the animation still works but the stirrer will be jumpy, lines will appear in the heated liquid and the falling drops will not be of realistic shape.

This concludes this series. A 5 1/4 disk with source code and BASIC listings of all programs listed in these five articles is available from the author for $10.

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