Building Your Own Tools: The-do-Command

3

You now know enough about presenting material to the student to be able to make attractive displays. You will be able to do even more when you learn how to tell PLATO to calculate complicated displays for you. Before discussing how to do calculations we will pause to introduce an extremely important concept, the “subroutine”, which is fundamental to all aspects of authoring. We will start off by applying the concept of a subroutine to certain display problems.

To introduce the use of subroutines, consider the problem of placing some standard message on several of your main lesson pages. For example, in the many units where you make help available to the student (if he presses the HELP key) you might like to advertise this fact by placing this display at the bottom of the page:

$\boxed{\text{HELP is available}}$

The corresponding TUTOR statements might be:

```
at      3123
write   HELP is available
box     3Ø22;3141
```

It would be tedious to copy these statements into every unit where they were required. Moreover, if you decided later to move this to the upper right corner of the screen, you would have to find all occurrences of this and change all of them. There is a way around these difficulties, and in later work we will find further important advantages to the method. Suppose we write a "subroutine" (a unit to be used many times as needed):

```
unit    helper
at      3123
write   HELP is available
box     3ø22;3141
```

Where we need to show this message, we need only write the statement:

```
do   helper
```

This statement attaches unit "helper" to the present unit. It is as though we had inserted the contents of unit "helper" at the point where we say "do helper". Now, instead of a dozen copies of the display statements we have only one, plus a dozen -do- commands. The -do- command may appear anywhere in a unit. The location of each -do- command will determine when the associated display appears on the screen in relation to your existing display material. All these displays may be changed by simply changing the subroutine unit! You do not need to change the -do- statements, instead change unit "helper" which they all use.

The use of -do- improves the readability of a TUTOR lesson. When you see "do helper" anywhere in your lesson you recognize at a glance what it is for. The contents of unit "helper" might contain a large number of statements which would clutter up your other units, and decrease readability, if these statements appeared directly in each unit.

Let's consider another use. Suppose we wish to draw a "Cheshire cat" which fades to a smile as Alice watches. We want to draw a cat face made up of the smile plus all the rest of the face, then erase everything but the smile. Here is an effective way of doing it. (See Figures 3-1a and 3-1b.)

```
unit     Alice
at       512
write    Watch the Cheshire cat!
do       cat
catchup           $$ wait for cat to be drawn
pause    4        $$ then pause 4 seconds
mode     erase
do       face
mode     write
at       3ø12
write    See the smile?
```

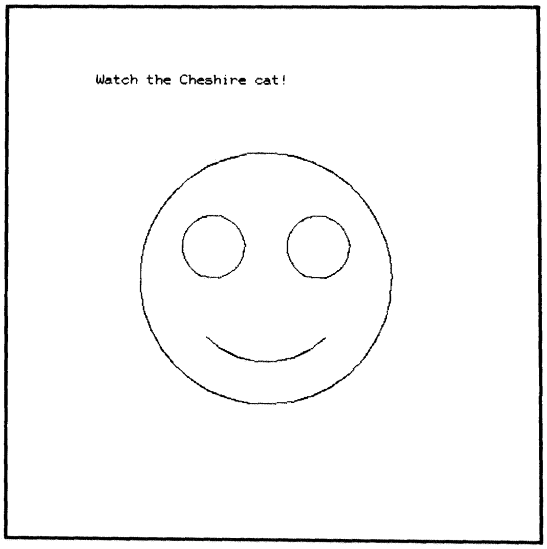

Fig. 3-1a.

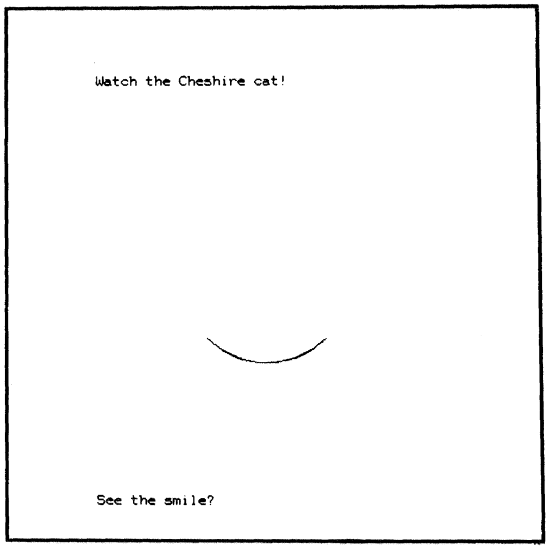

Fig. 3-1b.

We will need some units to use as the subroutines:

```
unit    cat
do      face
do      smile
*
unit    face
at      25Ø,25Ø
circle  12Ø          $$ outline
at      2ØØ,28Ø
circle  3Ø           $$ eye
at      3ØØ,28Ø
circle  3Ø           $$ eye
*
unit    smile
at      25Ø,25Ø
circle  8Ø,225,315   $$ smile—arc goes from 225° to 315°
```

Note that unit "Alice" does unit "cat", which in turn does units "face" and "smile". TUTOR permits you to go ten levels deep in -do-s. Here we have gone only two levels deep. Note that unit "smile", on its own, is a useful subroutine and might be done whenever just the smile is desired.

To summarize, we can build useful tools by constructing "subroutines" (units which may be done from many places in the lesson). The liberal use of -do- improves readability, reduces typing, and facilitates revising the lesson. This last point is particularly important when there is a "bug" (unknown error) in the lesson. Debugging becomes much simpler because of the modular nature of subroutines, and because a lesson which uses -do- extensively has its critical control points well localized.