

More on Creating Displays

Particular attention should be paid to the question of how to display text and line drawings to the student. Good or poor displays of material in a lesson can make the difference between a successful or unsuccessful lesson. Imaginative use of graphics, including animations (moving displays), will capture the attention of the student and transmit your message much more efficiently than would mere text. You have already seen how to write text and draw figures by using the `-at-`, `-write-`, and `-draw-` commands. This chapter will discuss how to achieve finer control over screen positions, how to draw circles and circular arcs, how to display large-size text and write at an angle, and how to erase portions of the screen. The ability to erase a portion of the screen makes it possible to create animated displays.

Coarse Grid and Fine Grid

It is convenient to specify a line number and character position for displaying text. We have seen that the TUTOR statement `"at 1812"` instructs PLATO to display information starting on the 18th line at the 12th character position. Line 1 is at the top of the screen and line 32 is at the bottom. Each line has room for 64 characters, with character position 01 at the left and character position 64 at the right. This numbering scheme is called the "coarse grid" or "gross grid".

Sometimes it is necessary to position text or draw a figure with finer control than is permitted by the coarse grid. The PLATO screen consists of a grid of 512 by 512 dots, and the position of any of these quarter-million dots can be specified by giving two numbers—the number of dots from the left edge of the screen (often called “x”) and the number of dots up from the bottom of the screen (often called “y”):

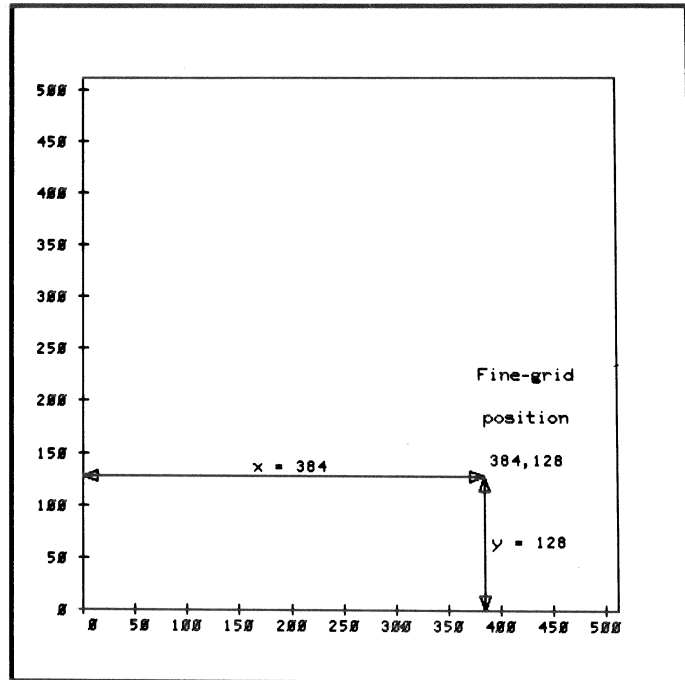


Fig. 2-1.

The position shown would be referred to as the “fine grid” location “384,128” in an -at- or -draw- statement. This position is equivalent to the coarse grid location 2449 (line 24, character position 49). As an example, consider the following unit:

```

unit double
at 384,128
write DOUBLE WRITING
at 385,129
write DOUBLE WRITING
    
```

This unit would write “DOUBLE WRITING” twice, displaced horizontally and vertically by one dot, which looks like this:

DOUBLE WRITING

Fig. 2-2.

(Greatly enlarged.)

The `-draw-` command permits mixing the two numbering schemes:

```
draw 1215;1225;120,240;1855
```

This means “draw a straight line from 1215 to 1225, draw a second straight line from there to (120,240), then draw a third straight line from there to 1855”. Note that each point, whether expressed in coarse grid or fine grid, must be set off by a semicolon.

The `-box-`, `-vector-`, and `-circle-` Commands

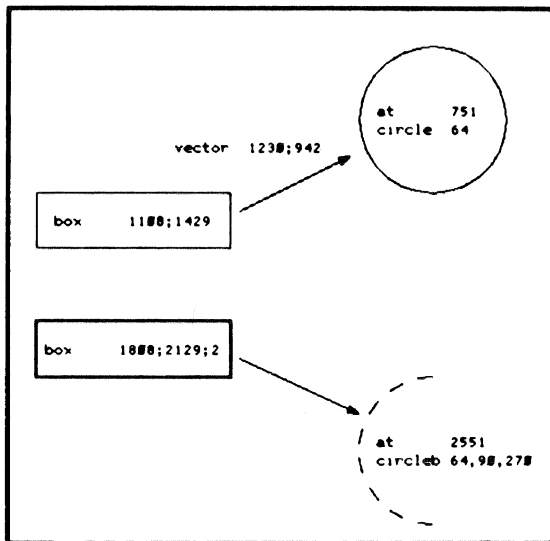


Fig. 2-3.

Figure 2-3 illustrates how rectangular boxes are often drawn as part of a display. Although such boxes can be drawn using the `-draw-` command, it is even more convenient to use a `-box-` command, since you merely give two corners of the box. For example:

```
box 1215;1835
```

is exactly equivalent to:

```
draw 1215;1235;1835;1815;1215
```

Fine grid coordinates can also be used for the corners of the box. The sides of the box can be made thicker for additional emphasis. For example, “box 1215;1835;2” will draw a box with sides two dots thick.

Another frequently drawn object is a “vector”—a line with an arrowhead used to point out something on the screen. The statement “vector 512;920” will draw a line from 512 to 920, with an arrowhead added at the 920 end. Fine grid coordinates can be used. The size of the arrowhead in relation to the line can be controlled by adding another number. For example, “vector 512;120;6” will show an arrowhead about half as large as normal. Making the arrowhead size *negative* draws an “open” rather than a “closed” arrowhead.

Circles are drawn by specifying a center with an `-at-` command, then using a `-circle-` command to specify the radius (as a number of dots):

```
at      1215
circle  50
circle  75
```

This will draw two circles whose radii are 50 dots and 75 dots long, centered at screen location 1215. Notice that the screen position is restored to the center of the circle after drawing a complete circle.

A portion or arc of a circle can be drawn by specifying starting and ending angles, as in “circle 125,0,45”, which will draw a 45-degree circular arc, starting at 0 degrees (0 is “east” or “horizontally to the right”; and 360 degrees is again “east”). After drawing an arc, the screen position is left at the end of the arc rather than at the center of the circle.

The `-circleb-` command is just like `-circle-`, but it draws a broken or dashed circle or circular arc.

The basic line-drawing commands (`-draw-`, `-box-`, `-vector-`, and `-circle-`) are used together to build complicated drawings.

Large-size Writing: `-size-` and `-rotate-`

It is possible to display text in larger than normal size, and even write at an angle. This is particularly useful in showing an eye-catching title on a page. Here is a sample display with the corresponding TUTOR statements. The “\$\$” permits a comment to appear after a tag.

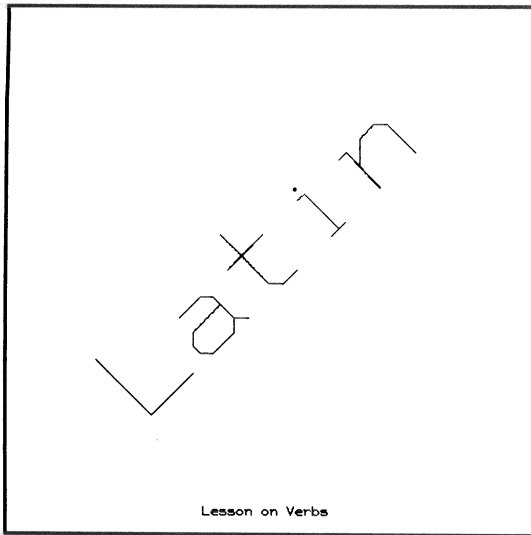


Fig. 2-4.

```

unit    title
size    9.5    $$ text 9.5 times normal size
rotate  45     $$ text rotated 45 degrees
at      2519
write   Latin
size    0      $$ return to normal writing
rotate  0
at      3125
write   Lesson on Verbs
    
```

For technical reasons the large-size writing comes on the screen much more slowly than does normal text, but the speed is adequate for short titles. Use “size 0” to return to normal writing. Normal writing is unaffected by -rotate-. However, you may use “size 1” if you wish to rotate standard size text. Size 1 writing appears at the same slow speed as larger writing (about 6 characters per second, or 60 words per minute). Only size 0 writing is rapid (180 characters per second, or 1800 words per minute).



BE SURE TO RETURN TO SIZE 0!! If you forget to place a “size 0” statement after the completion of the special writing, all of your text will be written slowly (and possibly rotated). It is also good practice to say “rotate 0”, so that the next time you use “size” the rotation will be through 0 degrees unless stated otherwise.

You can magnify the width of the characters differently from the height. For example, “size 2,5” will make the characters twice as wide and five times as high as they are normally.

Because “sized” writing is slow, it should be used only for special effects. It should be avoided on pages which are seen repeatedly, such as a table of contents for a lesson, because the student will be irritated by the enforced wait. In such cases, it is better to achieve emphasis by other, faster techniques, such as drawing a box around a heading written in “size 0”.

Animations (Moving Displays): -erase- and -pause-

An animated display can be created by repetitively displaying some text, pausing, erasing the text and rewriting it in a new position on the screen. Here is a unit which will show two balloons floating upwards. The unit is split in order to show the changes in the display. (See Figures 2-5a through 2-5c.)

	unit	balloons	
	at	3020	
	write	Watch the balloons go up!	
	at	250,100	
	write	00	\$\$ use 00 for balloons
	pause	1.5	\$\$ suspend processing for 1.5 seconds
	at	250,100	
	erase	2	\$\$ erase two characters
	at	250,150	\$\$ reposition 50 dots higher
	write	00	
	pause	1.5	

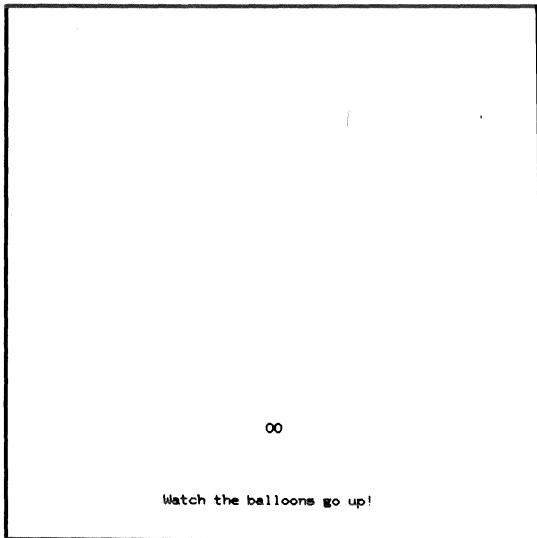


Fig. 2-5a.

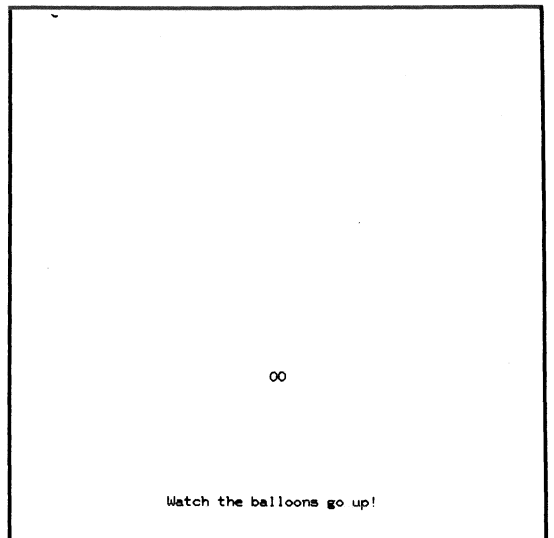


Fig. 2-5b.

```

at      250,150
erase   2
at      250,200
write   00
pause   1.5
.
.
.

```

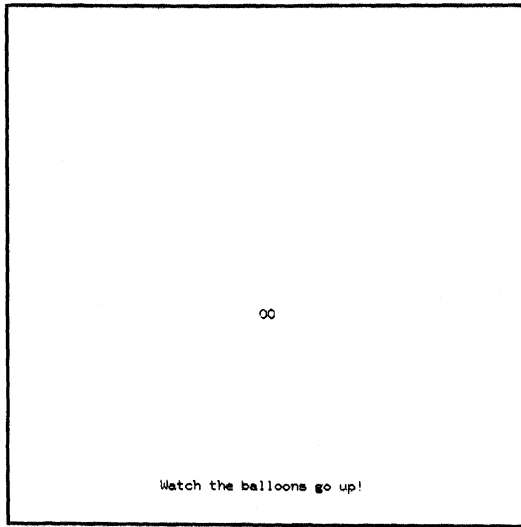


Fig. 2-5c.

The statement "erase 2" selectively erases two character positions without disturbing the rest of the screen. In particular, the text "Watch the balloons go up!" will stay on the screen.

There are other forms of the -erase- command. The statement "erase 12,3" will selectively erase a block of 12 character positions on three consecutive coarse grid lines. The statement "erase" with no tag will erase the entire screen instantaneously. The same full-screen erase normally takes place automatically upon moving to a new main unit.

-pause-, -time-, and -catchup-

The -pause- statement with a tag in seconds suspends processing for the specified amount of time. If the tag is omitted, TUTOR waits for the

The TUTOR Language

student to strike a key, any key, rather than wait a specified amount of time. This form is particularly suitable in more complicated situations where the student may want to study each step before proceeding. Here is an example:

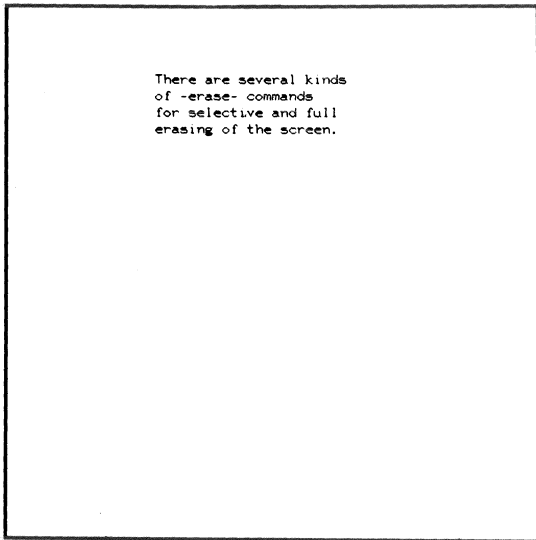


Fig. 2-6a.

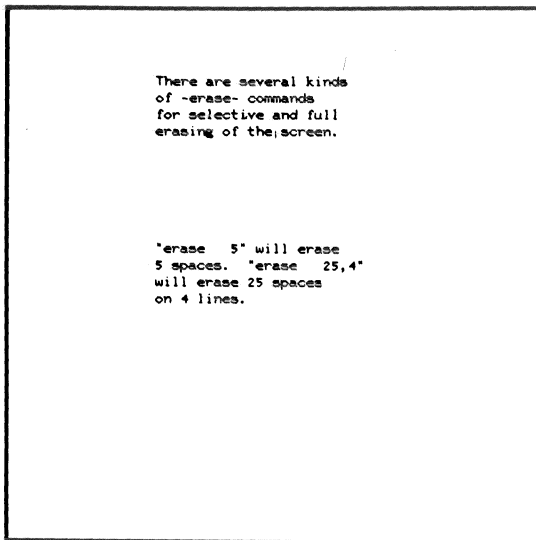


Fig. 2-6b.

unit discuss
at 520
write There are several kinds
 of -erase- commands
 for selective and full
 erasing of the screen.



pause
at 1520
write ""erase 5" will erase
 5 spaces. "erase 25,4"
 will erase 25 spaces
 on 4 lines.

pause
at 2520
write An -erase- command
 with a blank tag
 will erase the whole
 screen.


```

There are several kinds
of -erase- commands
for selective and full
erasing of the screen.

"erase 5" will erase
5 spaces. "erase 25,4"
will erase 25 spaces
on 4 lines.

An -erase- command
with a blank tag
will erase the whole
screen.

```

Fig. 2-6c.

Each time the student presses a key to move past the `-pause-` command, more text is added to the screen. This prevents the student from feeling overwhelmed by too much text all at once. Each new paragraph is added only when the student signals by pressing a key that he or she wants to go on. On the other hand, this structure leaves the earlier paragraphs on the screen so that the student can look back to review. If the `-pause-` commands were replaced by `-unit-` commands, each paragraph would reside in a separate main unit. When the student presses NEXT to move on to the next main unit, the screen is completely erased to make room for the next display. This would accomplish the objective of letting the student control the rate of presentation of new material but would not leave the earlier paragraphs on the screen for review and comparison.

It is inadvisable in this application to use `"pause 15"` rather than `"pause"`, since the student would have no control over the presentation rate. Any time delay you choose will be too fast for some students and too slow for others. A timed `-pause-` is mainly useful for animations. Sometimes it is appropriate to move on after a long time if the student hasn't pressed a key. This can be achieved with a `-time-` command:

```

.
time 30
pause
.
.

```

The “time 30” statement will “press the timeup key” after 30 seconds, so that if the student does not press a key, TUTOR will. The student can move on sooner by pressing a key before then. However, this is not possible if you use “pause 30”.

To summarize, there are three types of -pause- situations:

- 1) pause n pause n seconds whether
keys are pressed or not
- 2) pause wait for any key
- 3) time n } wait for any key or n seconds
pause }

Occasionally, you might want to send several seconds worth of output to the student’s screen, then pause two seconds, then add something else. If you write several seconds of display including text and drawings which take several seconds to paint on the screen, followed by:

```
pause 2
write More text. . . .
```

you will not get the desired effect because TUTOR will add “More text . . .” right after the initial material headed toward the terminal (since the “pause 2” ends before the initial display is finished). The student will see no gap between the first and second parts of the display. The problem can be solved with a -catchup- command:

```
.
.
.
catchup
pause 2
write More text. . . .
```

The -catchup- command tells TUTOR to let the terminal “catch up” on its work up to that point before continuing. *Then* you pause an additional two seconds, and you get the desired effect.

The -mode- Command

The -erase- command may be used to erase blocks of character positions or the whole screen. However, something else is needed for selectively erasing line drawings created with -draw- and -circle- statements. The PLATO terminal can be placed in an erasure mode in which the terminal interprets all display instructions as requests to erase rather than to light up the corresponding screen dots. This is done with the -mode- command:

```

unit    modes
at      2517
write   Selective erase of a figure
draw    1210;2010;2050;1210  $$ triangle
pause   $$ wait for a key
mode    erase
☞ draw  1210;2010;2050      $$ part of the triangle
mode    write
at      510
write   One line left.
    
```

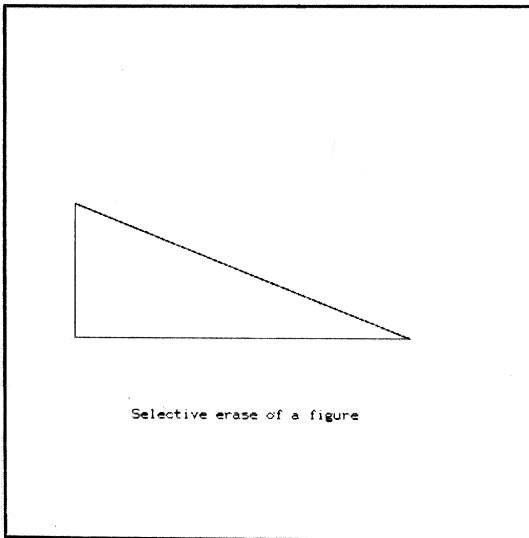


Fig. 2-7a.

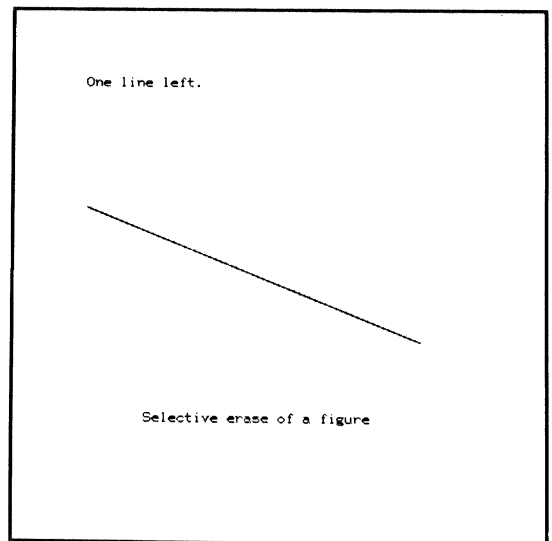
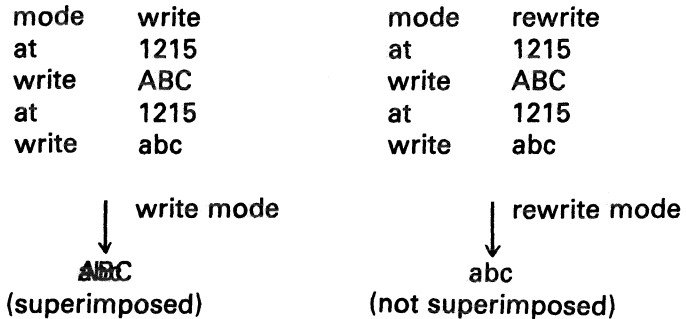


Fig. 2-7b.

The “write” mode is the normal display mode. Be sure to specify “mode write” when you are through with “mode erase”, or all further writing in that unit will be invisible!

In the standard mode (“write”) it is possible to superimpose or overstrike text with another -write- statement. If, however, a “mode rewrite” statement is executed, the second -write- statement will erase the previous text as it writes the new text, and there will be no superposition. Compare these sequences in write and rewrite modes:



In the rewrite case the second -write- statement wipes out the 3-character area as it writes the new information. Each character area is 8 dots wide by 16 dots high. This determines the number of rows and columns in coarse grid. In the coarse grid, $(512/8)=64$ characters fit across the screen, and $(512/16)=32$ lines of characters fill the screen vertically.

The statement “erase 2” is actually equivalent to:

```

mode  rewrite
write (two spaces)
mode  (previous mode)
                
```

Writing spaces (blank characters) in rewrite mode wipes out an entire character area.

The balloon animation in Figures 2-5a through 2-5c could have been written:

```

.
.
.
at     250,100
write  00
pause  1.5
mode   erase
at     250,100
                
```

```

write 00      $$ instead of "erase 2"
mode write
.
.
.

```

This form would be different from the form using "erase 2" if there were other screen dots lit in this area. The form which uses "erase 2" completely erases two character positions while "write 00" in the erase mode erases only the dots that make up the letters "00" without disturbing neighboring dots.

Automated Display Generation

It should be mentioned that an author working at a PLATO terminal can use a moving cursor to design a display involving text, line figures, circles and arcs. The PLATO system then automatically creates corresponding TUTOR statements which would produce that display. The author can alter these statements, convert them back into a display, and add to or alter the resulting display. This facility makes it unnecessary in most cases to worry about the details of screen positions. Here is an example of such operations:

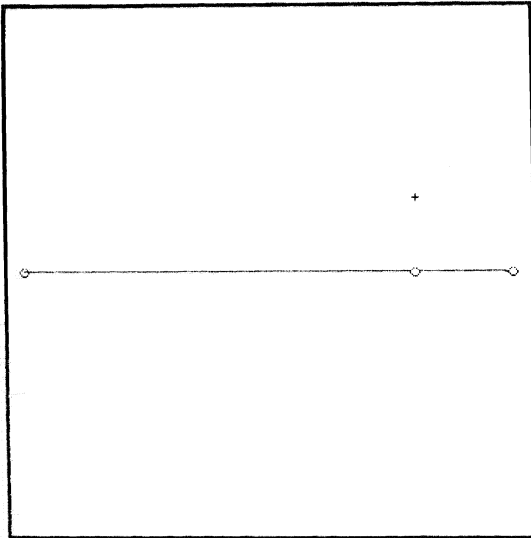


Fig. 2-8. Move the cursor (the "+") to draw the road and to mark the ends of the tree trunk.

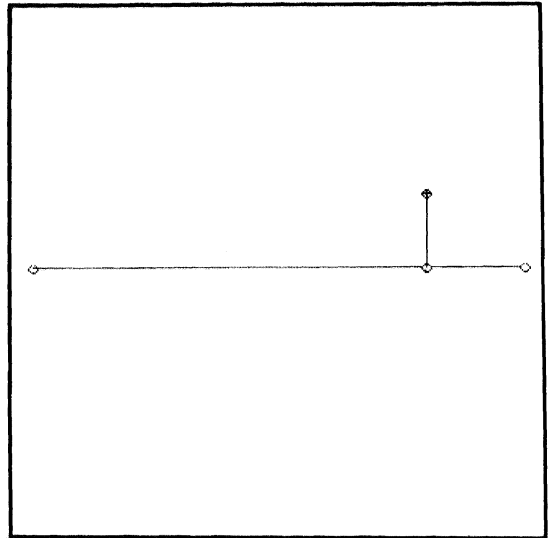


Fig. 2-9. Draw the tree trunk.

The TUTOR Language

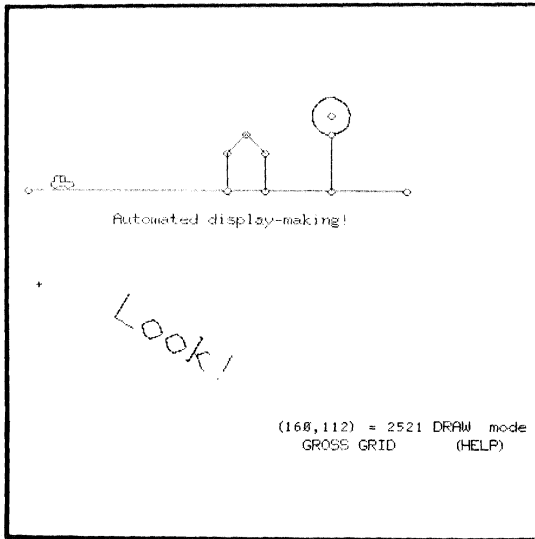


Fig. 2-10. Specify a circle for the top of the tree. Draw the house. Place text of various kinds on the screen. (The car uses special characters.)

```
unit display
draw 1812;1852;skip;1844;1544
at 344,288
circle 16
draw 1837;1637;1535;1633;1833
at 104,225
write (C)
at 2021
write Automated display-making!
size 3
rotate -30
at 2521
write Look!
size 8
rotate 0
```

Fig. 2-11. PLATO automatically generates TUTOR statements corresponding to the desired display.

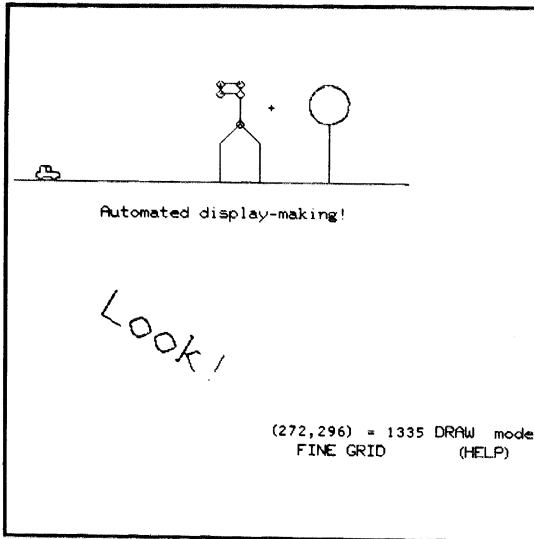


Fig. 2-12. Recall the display and add a flag to the house.

```

unit display
draw 1812;1852;skip;1844;1544
at 344,288
circle 16
draw 1837;1637;1535;1633;1831
at 184,225
write
at 2821
write Automated display-making
size 3
rotate -38
at 2521
write Look!
size 8
rotate 8
draw 1535;1335;1333;256,296;272,296
    
```

Fig. 2-13. PLATO appends a -draw-state-ment corresponding to the flag.

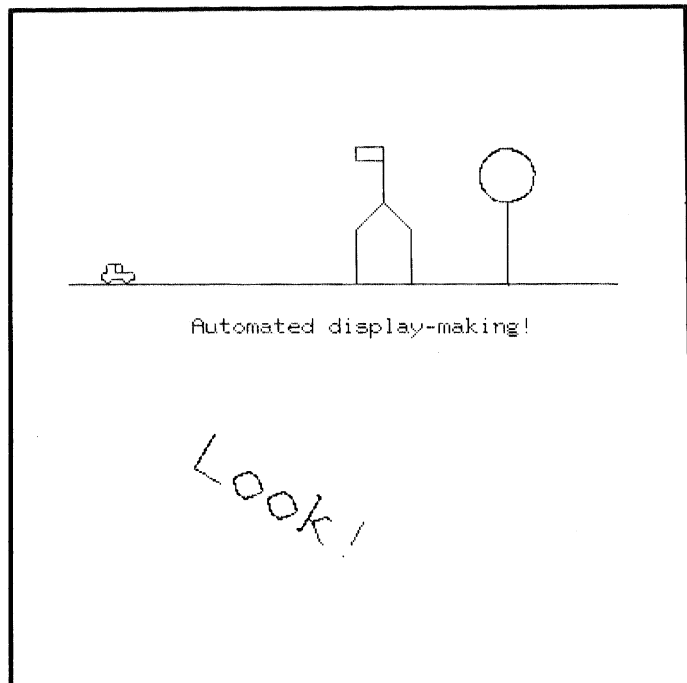


Fig. 2-14. Final result. The illustrations in this book were created by these techniques. The screen displays were photographed.