run, of course, until the subprocedures have been written or dummy routines provided in their place. To show how this works, let's consider how the dog-drawing program was constructed.

The DOG procedure was written first, even though none of its component procedures yet existed. We then wrote each of the shape-drawing procedures separately. These were followed by the positioning procedures. Each time a new procedure was written, DOG was run to ensure that everything fitted together properly. When LOGO came to a MOVE procedure that had not been written it stopped with an error message. However, it was easy to tell from the drawing whether everything up to this point was correct, or if there was an error in the last MOVE procedure.

Our set of procedures demonstrates another important point — each of the shape procedures, and the DOG procedure itself, leaves the turtle state unaltered. That is, the turtle is at the same position with the same heading at the end of the procedure as it was before the procedure was run. Such procedures are said to be *state transparent.* Making procedures state transparent eases the task of putting procedures together to construct more complex drawings. Take the DOG procedure, for example: once the turtle is positioned we know that after drawing a piece the turtle will return to the position it was in when it started that piece. So we need know nothing about the internal workings of the procedures in order to put the pieces together. By making DOG state transparent, we make it easier to use this procedure as part of another — for example, we could draw a whole screenful of dogs.

LOGO WORKSPACE

By now you will have a fair number of procedures in the computer's memory — so let's take a closer look at LOGO memory organisation. LOGO's working memory consists of a list of *nodes* (each of five bytes). Once LOGO is loaded, you will have between 1,000 and 3,000 of these, depending on the machine you use. As procedures are defined, these nodes are used up. Other nodes may be used as procedures are run or if variables (to be discussed later in the course) are used.

The procedures you have defined constitute your *workspace.* You can see which procedures are held in the workspace by entering POTS (for PRINTOUT TITLES). To look at an individual procedure, use PO (for PRINTOUT) — for example, PO SQUARE. If a procedure is no longer required, workspace can be freed by using ERASE — the command ERASE SQUARE would remove the procedure called SQUARE from memory. Erasing a procedure releases the nodes used. LOGO will mark these nodes, but will not yet add them to the list of free nodes; instead it will continue working with its present free nodes list until all of these have been used up. It will then go through its memory, gathering up all the nodes that have been released and using these to form a new list of free nodes. This process is referred to as *garbage collection* and is the reason why LOGO seems to hesitate for a second or two from time to time.

SAVING PROCEDURES

In order to make permanent records of your procedures on disk, you must save the workspace as a file. Using MYPROCS as an example file name, you would type SAVE "MYPROCS (note the quotation marks before, but not after, the file name). The workspace itself will not be affected by this. The file may be loaded with READ "MYPROCS. This causes the procedures in the file to be defined and added to the current workspace. If a procedure is defined with the same name as a procedure already held in the workspace, then the new definition replaces the earlier one.

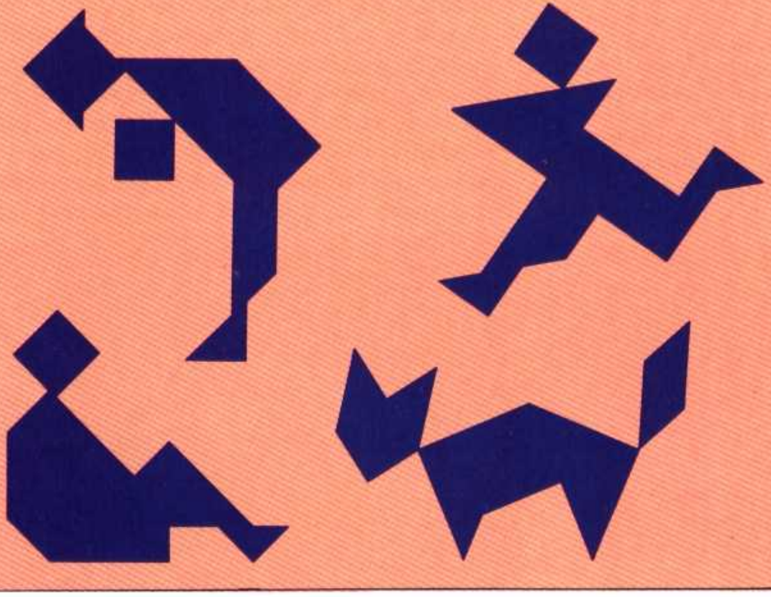
Other useful disk-handling commands are CATALOG and ERASEFILE. CATALOG gives a list of all the files on the disk, and ERASEFILE "MYPROCS would erase the MYPROCS file from the disk. Cassette-based versions of LOGO use different commands — the relevant manual should be consulted for these.

Abbreviations

ERASE	ER
PRINTOUT	PO
PRINTOUT TITLES	POTS

Procedure Problems

1) Write procedures for the other tangram shapes shown. (You will first have to solve the puzzle of how to construct the shape from the different pieces, of course!)
2) Write a procedure to draw a 'house' (simply an equilateral triangle above a square).
3) Write a procedure to draw a five-by-five board of squares.
4) Rewrite the procedure used earlier to draw a six-pointed star so that it uses subprocedures.


LIZ DIXON

Logo Flavours

In all LCSI versions, procedure names must be prefixed by a quotation mark if they are inputs to PO or ERASE: for example, PO "SQUARE and ERASE "SQUARE.

To read a disk file use LOAD "MYPROCS.

Versions supporting cassette tapes or microdrives have somewhat different commands from those for disks. You should consult the relevant technical manual.