

Washington University in St. Louis

## Washington University Open Scholarship

---

All Computer Science and Engineering  
Research

Computer Science and Engineering

---

Report Number: WUCS-86-04

1986-03-01

### Show and Tell User's Manual

Peter McLain and Takayuki Dan Kimura

The purpose of this report is to introduce essential features of the Show and Tell Language system to those computer users who are already familiar with some high-level programming language such as FORTRAN, BASIC or PASCAL. This manual is not intended for school children. Some familiarity with the Macintosh user interface and the MacPaint application program is assumed. It is also assumed that the Show and Tell application program disk and the Sample program are available to the reader. The basic programming concepts in Show and Tell are introduced in Chapter Three. The reader may find it easier to... [Read complete abstract on page 2.](#)

Follow this and additional works at: [https://openscholarship.wustl.edu/cse\\_research](https://openscholarship.wustl.edu/cse_research)



Part of the [Computer Engineering Commons](#), and the [Computer Sciences Commons](#)

---

#### Recommended Citation

McLain, Peter and Kimura, Takayuki Dan, "Show and Tell User's Manual" Report Number: WUCS-86-04 (1986). *All Computer Science and Engineering Research*.  
[https://openscholarship.wustl.edu/cse\\_research/839](https://openscholarship.wustl.edu/cse_research/839)

Department of Computer Science & Engineering - Washington University in St. Louis  
Campus Box 1045 - St. Louis, MO - 63130 - ph: (314) 935-6160.

## Show and Tell User's Manual

Peter McLain and Takayuki Dan Kimura

### Complete Abstract:

The purpose of this report is to introduce essential features of the Show and Tell Language system to those computer users who are already familiar with some high-level programming language such as FORTRAN, BASIC or PASCAL. This manual is not intended for school children. Some familiarity with the Macintosh user interface and the MacPaint application program is assumed. It is also assumed that the Show and Tell application program disk and the Sample program are available to the reader. The basic programming concepts in Show and Tell are introduced in Chapter Three. The reader may find it easier to start with Chapter Three, then come back to Chapter One and Chapter Two.

**SHOW AND TELL USER'S MANUAL**

**Peter McLain and Takayuki Dan Kimura**

**WUCS-86-04**

**March 1986**

**Department of Computer Science  
Washington University  
Campus Box 1045  
One Brookings Drive  
Saint Louis, MO 63130-4899**

**This research was funded by Computer Services Corporation (CSK) of Japan. Show and Tell is a trademark of Computer Services Corporation (CSK).**

## Table of Contents

### Chapter One: Tour of Show and Tell

Starting Show and Tell	1
Sample Drawers	2

### Chapter Two: Using Show and Tell

§ 2.1 Editing in Show and Tell	12
Drawing boxes and arrows	12
Changing boxes and arrows	14
§ 2.2 File handling	16
Drawers	16
Puzzles	17
§ 2.3 Database	19
Schema	19
Solutions	19
Query	20

### Chapter Three: Programming in Show and Tell

§ 3.1 Computer Programs	21
§ 3.2 Puzzles	21
§ 3.3 Defining problems with puzzles	22
Computation problems	22
Simple problems	22
Iteration	24
Database problems	32
Solutions / Schema	32
Query	34

### Chapter Four: Reference

§ 4.1 The Show and Tell Environment	35
§ 4.2 The elements of Show and Tell	36
§ 4.3 Editing icons	39
§ 4.4 Menus	43
§ 4.5 System Drawers	47

Show and Tell is a trademark of Computer Services Corporation (CSK)  
Macintosh™ is a trademark of Apple Computer®  
SmoothTalker® is a trademark of First Byte®

## Preface

The purpose of this report is to introduce essential features of the Show and Tell Language system to those computer users who are already familiar with some high-level programming language such as FORTRAN, BASIC or PASCAL. This manual is not intended for school children. Some familiarity with the Macintosh user interface and the MacPaint application program is assumed. It is also assumed that the Show and Tell application program disk and the Sample program disk are available to the reader\*.

The basic programming concepts in Show and Tell are introduced in Chapter Three. The reader may find it easier to start with Chapter Three, then come back to Chapter One and Chapter Two.

A detailed description of the Show and Tell Language system is given in the following technical report:

Takayuki Dan Kimura, Julie W. Choi, and Jane M. Mack  
*A Visual Language for Keyboardless Programming*  
Technical Report WUCS-86-6, Department of Computer Science  
Washington University, St. Louis, MO 63130, March 1986.

---

\* The prototype Show and Tell system disks can be obtained from:  
Mr. K. Hikawa, Technical Manager  
Computer Services Corporation  
17F Sumitomo Building  
2-6-1 Nishi-Shinjuku  
Shinjuku-ku, Tokyo 160-91  
Japan  
Tel: (03) 344-1811.

## § 1.1 The Show and Tell Environment

### Starting a Show and Tell Session

This chapter is designed for those people new to the Show and Tell environment. The Show and Tell environment is the place that you will do your work. In it you can solve puzzles, edit old puzzles, or create new puzzles. The environment consists of features common to all Macintosh programs (e.g. pull down menus, scroll boxes, and windows) and features unique to the Show and Tell Language (e.g. puzzles, solutions, and box structures). We assume that you are already familiar with the first set of features. This chapter will introduce you to the second set. Show and Tell solves problems by completing puzzles that are written by you. In this section we will see some examples of puzzles and get a feel for how to use them in the Show and Tell environment.

Before you can use the Show and Tell Language, you will need the following :

- \* 512K Macintosh
- \* Macintosh External Disk Drive
- \* Show and Tell Program Disk
- \* A System Disk (a disk with the finder) that has at least 150K of free space.

**Note:** You may make your own Show and Tell startup disk by copying the two files labeled "iconRsrc" and "mainRsrc" onto a system disk. This will allow a quicker start up for Show and Tell. If you make your own startup disk, you do not need the 150K of free space.

To begin a Show and Tell session:

- \* Turn the Macintosh on.
- \* Put the system disk in the Macintosh internal drive and the Show and Tell program disk in the external drive.
- \* Open the Show and Tell disk by double clicking on the disk icon named 'Show and Tell'.
- \* Start the Show and Tell program by double clicking on the Show and Tell program icon.

In this tutorial, we will be concerned with three parts of the screen; the menu bar, the editing tools, and the editing window. The **menu bar** is at the top of the screen and contains the Show and Tell menus. Each menu has several related entries which are Show and Tell commands. The **editing tools** are in the column of icons along the left side of the screen. These tools are used to create and change (edit) puzzles. The **editing window** is the large blank window that occupies most of the screen. This window will contain the puzzle we will be working with. There are two parts to the editing window, the **background** and the **name area**. The background is the part of the window upon which the puzzle is shown. The name area holds the name of the puzzle.

## § 1.2 Sample Drawers

### Opening a Drawer

### Drawers

### Loading a Puzzle

The first several puzzles we will look at are all contained in a drawer named **simple**. In order to take a look at these puzzles, we will need to open the drawer that they are in.

To open the drawer:

\* Select **Open** from the **Drawers** menu.

A dialogue box is opened which contains the names of all the drawers on the disk. (You may have to click on the "Drive" button to select the disk with the sample puzzles) We want to open the drawer called 'simple'.

\* Double click on the word 'simple'.

Your screen should now have the 'simple' drawer opened (figure 1.1).

A drawer is the place that Show and Tell keeps puzzles when they are not in use. The 'simple' drawer contains ten puzzles. Each puzzle is represented by an icon in the drawer. The icon is the name of the puzzle and is used whenever we want to refer to or use the puzzle.

We will now look at the first puzzle in the drawer. To use a puzzle, we must move its icon into the current puzzle area (figure 1.1).


To load the simple puzzle:

\* Move the pointer to the 'simple' drawer and place it on the

 icon.

\* Drag the icon into the current puzzle area.

\* Release the button.

(You may also open the puzzle by double clicking on the  icon.)

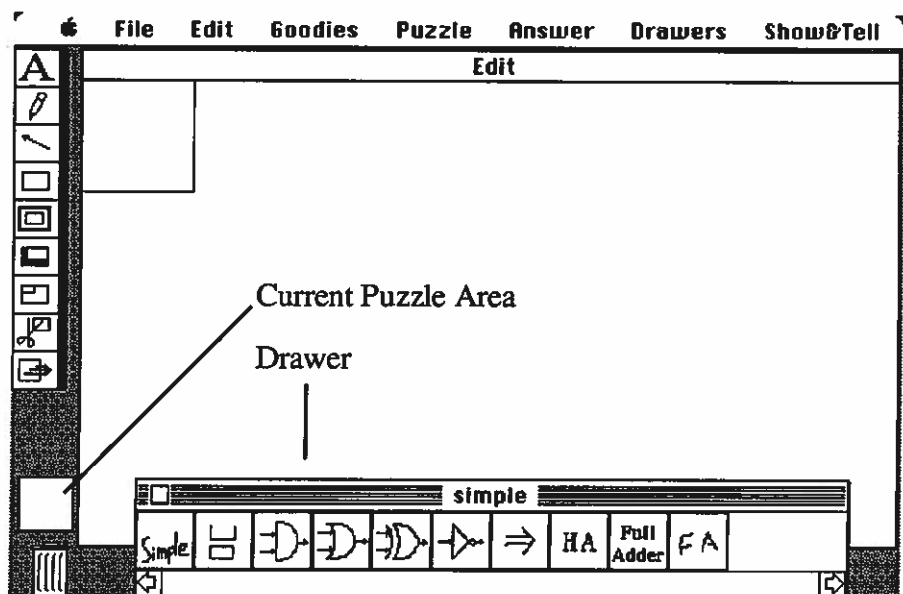


Figure 1.1

## The puzzle

Your screen should now look like figure 1.2. The puzzle is loaded and ready for use. The name of the puzzle appears in the upper left corner of the editing window. You will notice that it is the same as the icon you dragged into the current puzzle area.

This example will introduce three aspects of Show and Tell; how to solve puzzles, an introduction to inconsistency, and a brief look at how to interpret a puzzle.

There are three parts to this puzzle. Part of the puzzle solves the following equation:  $(2 * 3) + 3 = ?$ ; part of the puzzle asks Show and Tell if three is less than two; and part of the puzzle demonstrates the basic idea of inconsistency and how Show and Tell handles it.

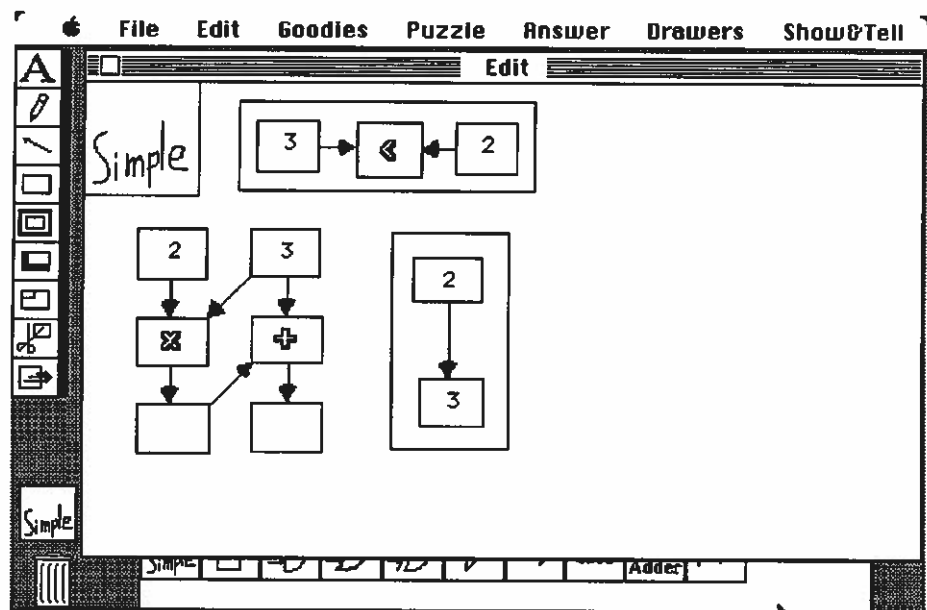


Figure 1.2

### Solving a puzzle

To solve the puzzle:

- \* Select **Solve** under the **Puzzle** menu.

Show and Tell has now solved or completed the 'simple' puzzle (figure 1.3). We can now see how Show and Tell solved each of the puzzle parts.

### Interpreting a puzzle

The answer to:  $(2 * 3) + 3$  is of course 9. Show and Tell has put the answer in the box pointed to by the "plus" box. The way this part of the puzzle works is straightforward; follow the arrows and do what is indicated in each box. 2 and 3 are given to the multiplication box which puts  $2*3$ , or 6, into the empty box. 6 and 3 are then given to the addition box and the answer is put into the remaining empty box. Show and Tell has filled in all of the blank boxes. We call this notion of filling-in **completion**.



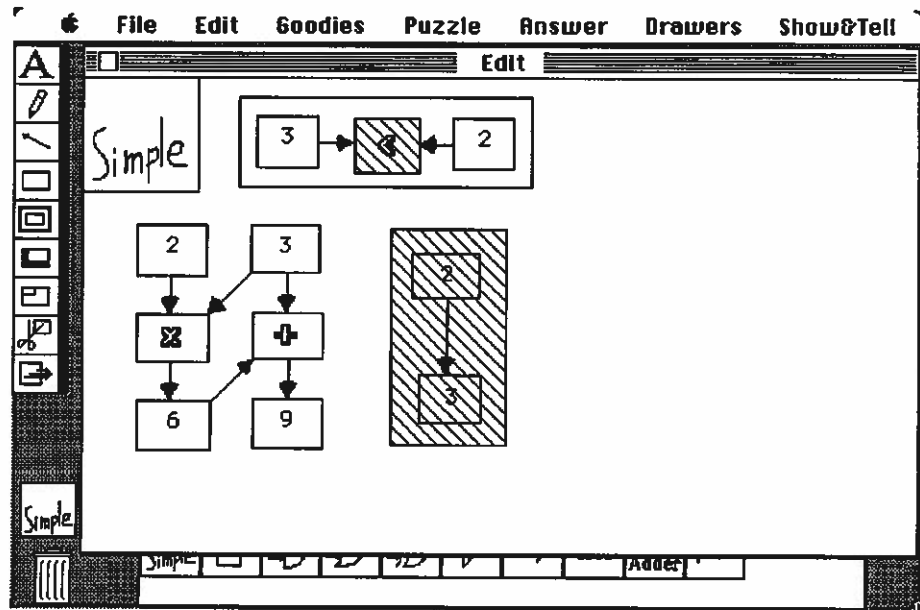


Figure 1.3

### Inconsistency

We also asked Show and Tell if three was less than two. Show and Tell answered our question by shading the "less than" box. This means that three is not less than two. If we had asked if two were less than three, Show and Tell would have left the "less than" box unshaded. Whenever Show and Tell comes across something that doesn't make sense, whenever there is an arrangement of boxes that results in an inconsistent state of affairs, Show and Tell will shade the the boxes to indicate which part of the puzzle is inconsistent. Part of the puzzle asserts that three is less than two. Show and Tell checked the assertion  $3 < 2$  and found it to be inconsistent with the definitions of 3, 2, and  $<$ .

Another example of inconsistency is shown in the last part of the puzzle. Here we were trying to give the value 2 to a box that already had 3 in it. Since the values of the boxes were not the same ( $2 \neq 3$ ) the box was marked inconsistent.

## The puzzle

### Open and Closed Boxes

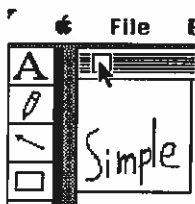




Figure 1.4

The next puzzle in the the drawer is called . The purpose of this puzzle is to illustrate the difference between the open and closed boxes. This example develops the idea of inconsistency by demonstrating how inconsistency may be used as a switch.

Load the  puzzle:

- \* Close the 'simple' puzzle by clicking in the close box in the top left corner of the editing window (figure 1.4).
- \* Move the  icon into the current puzzle area.

There are two parts to this puzzle (figure 1.5). The only difference between them is that in the upper part of the puzzle, the 3 --> 2 assertion is contained in a box with a dotted line (an open box), while in the lower portion the assertion is contained in a box with a solid line (a closed box). We know from the previous example, that Show and Tell will shade portions of the puzzle that are inconsistent. The open and closed boxes allow us to specify how much of the puzzle is shaded. A closed box keeps any inconsistency inside of it. An open box lets the inconsistency flow out into the surrounding environment.

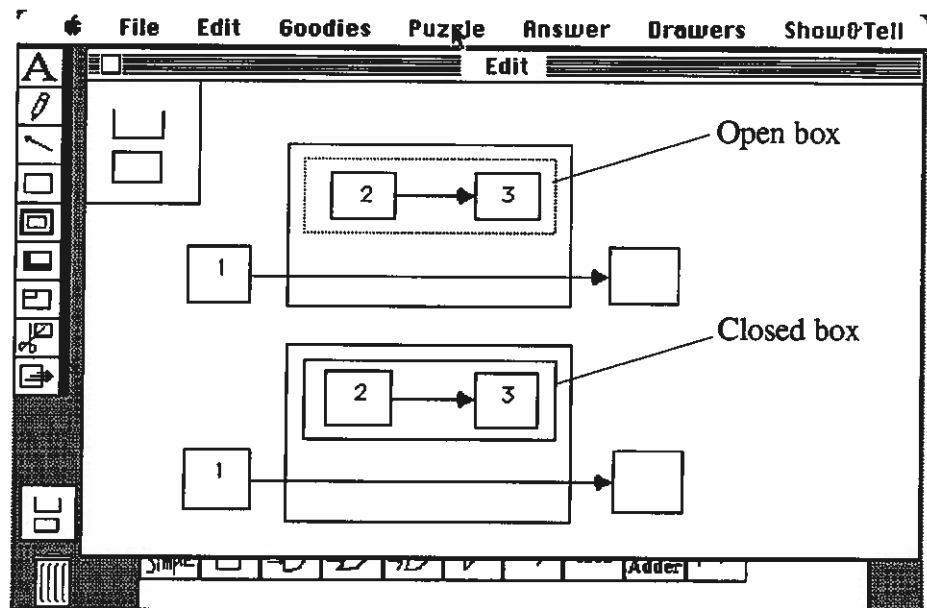


Figure 1.5

- \* Solve the puzzle by selecting **Solve** under the **Puzzle** menu.

## Shading Inconsistency

In figure 1.6 we can now see how the shading of the inconsistency is contained by the closed box in the bottom of the puzzle, but is allowed to flow out of the open box in the top of the puzzle. This puzzle also demonstrates how inconsistency can be used as a switch. No information can flow through inconsistency. In the top part of the puzzle, the arrow between the empty box and the box with a '1' in it goes through an inconsistent part of the puzzle. This causes the arrow to be "turned off" and hence the empty box at the right remains empty. In the bottom part of the puzzle, the inconsistency is not allowed to touch the arrow and so the value 1 is put into the box on the right.

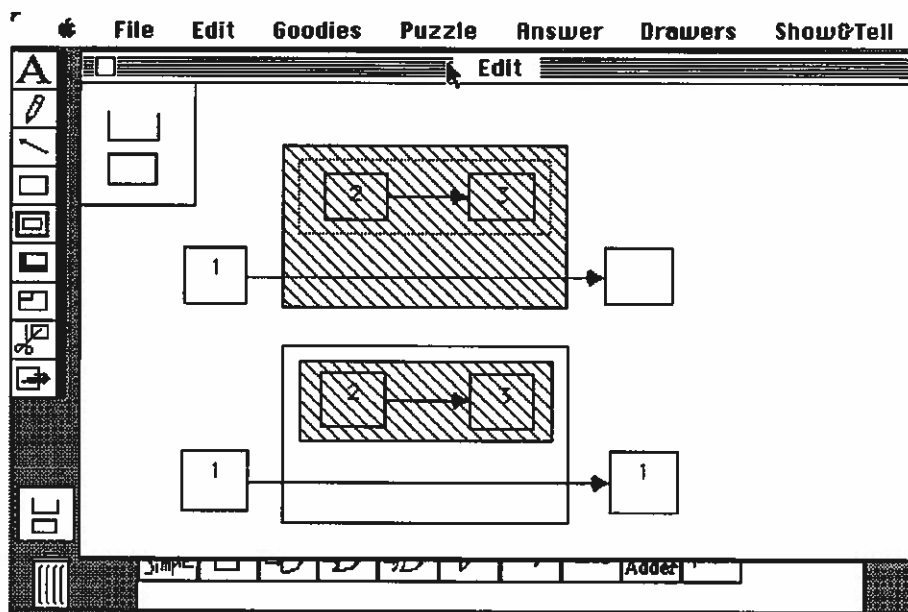




Figure 1.6

The  puzzle

- \* Close the puzzle by clicking in the close box.
- \* Load the next puzzle in the drawer.

This puzzle takes two inputs which are either '1' or '0' and produces the logical AND of the inputs. To run this puzzle, you must first type in the values you wish to AND.

- \* Move the pointer to the  icon (text tool) near the top left of the screen and click the button.
- \* Move the pointer inside one of the thick lined boxes and click the button.
- \* Type either a '1' or a '0' in the box.
- \* Fill in the other box with a '1' or a '0'.
- \* Solve the puzzle.

The solution to the puzzle should be '1' if both of the numbers you typed in were '1', and the answer should be '0' in all other cases. You may try other combinations of input values by selecting the text tool and then dragging it across the number you want to change. When the number is highlighted, you can then type in the new value and solve the puzzle.

The next three puzzles in the drawer define the logical operations of OR, EXCLUSIVE OR, and NOT. These puzzles are much like the AND puzzle we have just seen. The last two puzzles use these logic gates to simulate a binary half adder (HA) and a full adder (Full Adder). We do not have time to go through each of these examples in detail. The important point to get from the last two puzzles in the drawer is the use of previously defined puzzles as subroutines.

The animals drawer

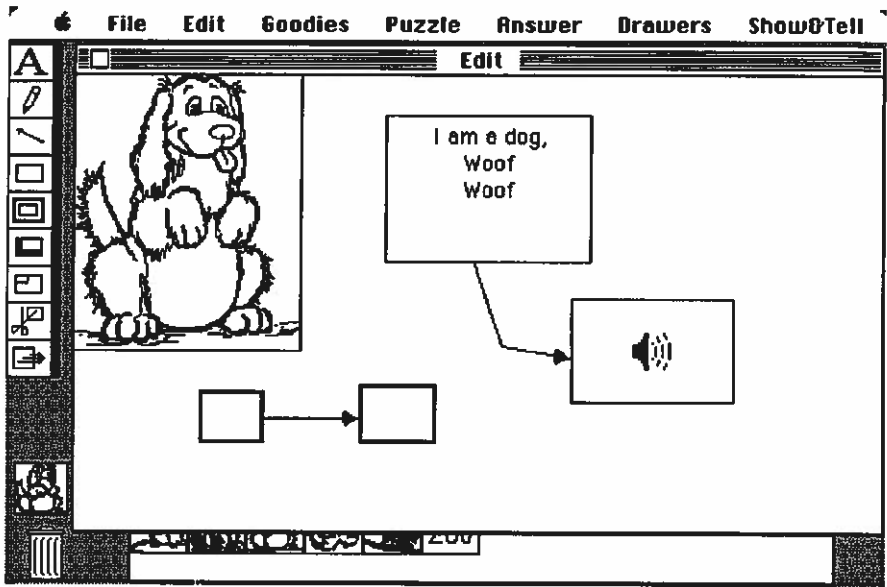


Figure 1.7

Open the drawer labeled 'animals'. This drawer contains six puzzles. The first five puzzles have the same structure. The last puzzle, 'Zoo', uses the other puzzles to give a tour of the zoo.

\* Load one of the first five puzzles (figure 1.7 shows the dog puzzle).

The name of this puzzle is the picture of the dog. This is a very simple puzzle that demonstrates one of Show and Tell's output features. The phrase in the box at the top is in a box with an arrow connecting it to the speaker icon. When the puzzle is solved, the phrase in the box will be spoken through the internal speaker\*\*.

\* Solve the puzzle.

The Macintosh now speaks for the dog. The other animals also speak. To hear each of the animals speak, load the 'Zoo' puzzle and solve it (figure 1.8). The animals will all speak in their turn.

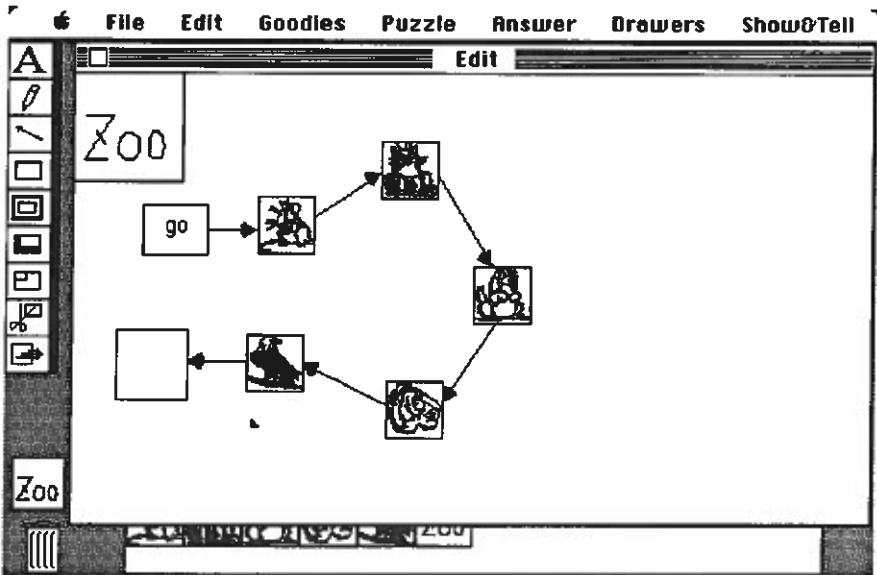


Figure 1.8

\*\* Show and Tell uses the SmoothTalker® synthesized speech driver from First Byte® to generate its voice output.

## Gcd Drawer

The third puzzle in the GCD drawer finds the greatest common divisor of two integers by Euclid's algorithm. The first two puzzles are used as subroutines. Show and Tell has multitasking capabilities which we will now demonstrate. We will have two puzzles being solved simultaneously on the screen (Show and Tell's multitasking is not limited to two tasks).

- \* Load the GCD puzzle.
- \* Place the mouse in the lower right hand corner of the editing window.
- \* Hold the mouse button down and drag the window until the puzzle just fits in it (figure 1.9).

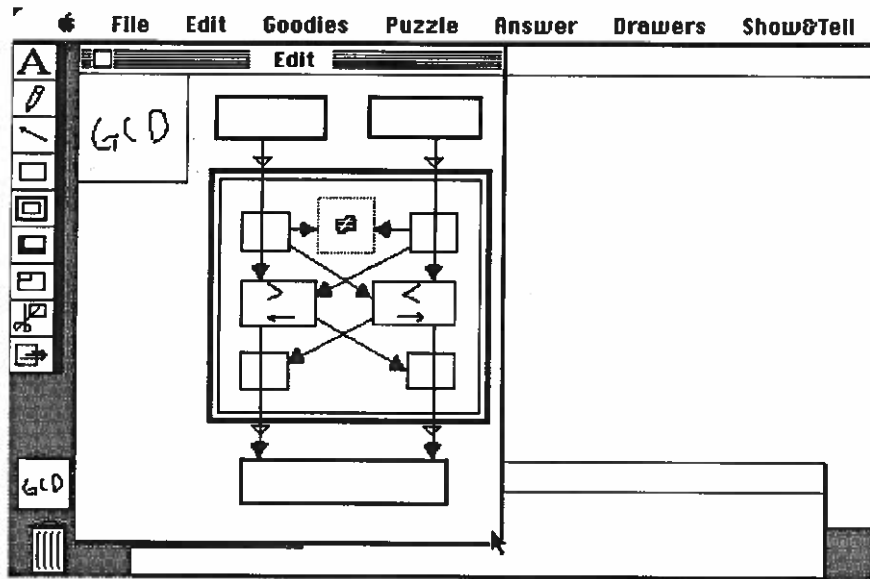


Figure 1.9

- \* Click on an exposed portion of the GCD drawer to activate it.
- \* Drag the GCD icon into the current puzzle area. (Load GCD again)
- \* Place the pointer on the move bar at the top of the new window and drag the new window to the right (figure 1.10).

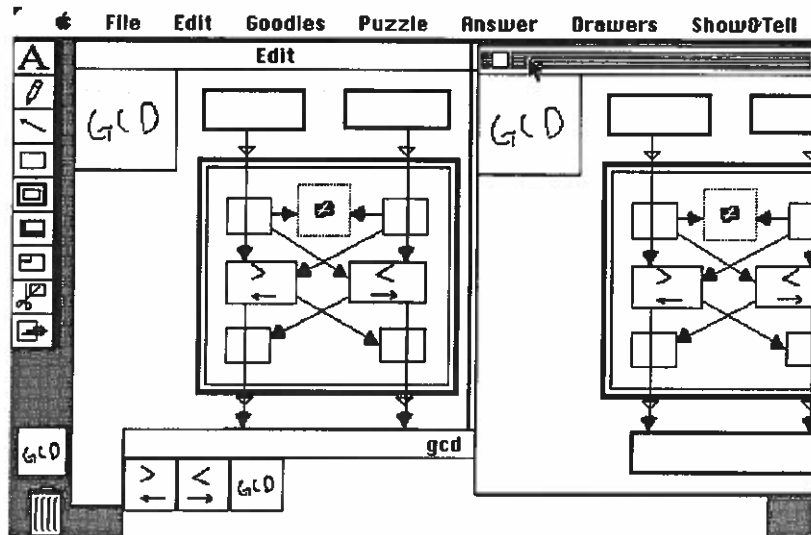


Figure 1.10

- \* Enter values into each of the GCD puzzles. (e.g. 123 and 672 for the first puzzle and 42 and 156 for the second puzzle (figure 1.11)).
- \* Click on the left hand window to activate it, then choose solve from the Puzzle menu.
- \* Now click on the other window and solve it.

Both puzzles should now be in the process of being solved simultaneously. If you entered the numbers we suggested (and you got the second puzzle going quickly enough) the second puzzle should be completed before the first one.

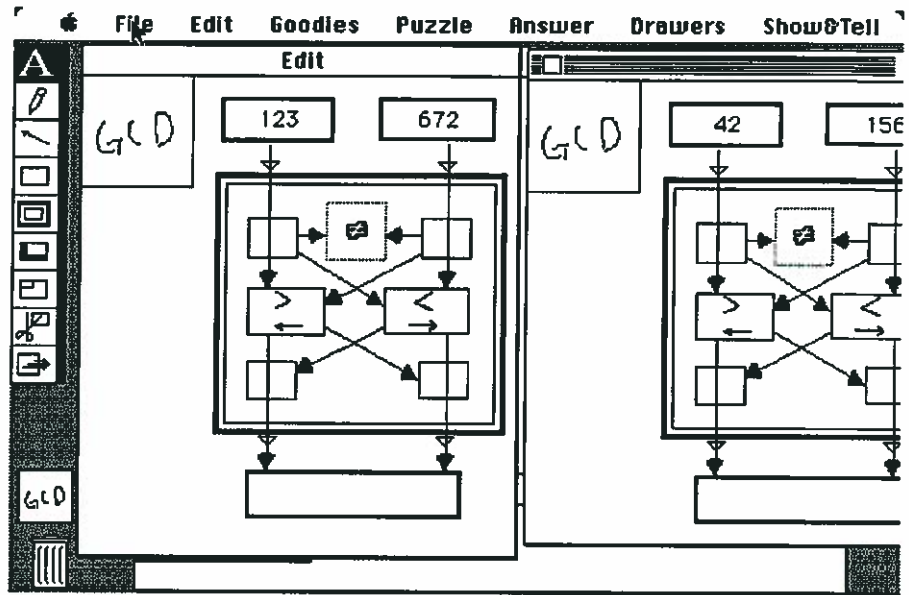


Figure 1.11

## STL Database Drawer

This example demonstrates some of the basic database capabilities of Show and Tell.

- \* Load the first puzzle in the drawer.

What appears on the screen is a database schema. The STL database holds the name, photo, and other information about each person that worked on the Show and Tell project.

- \* Select the text icon by clicking on it.
- \* Enter 'Jane Mack' (without quotes) into the box marked 'Name'.
- \* Select Find under the Answer menu.

The screen should now display Jane's database record.

To single step through the database (there are only 8 records):

- \* Select Buttons from the Answer menu.

The buttons now appear in the lower left of the window.

- \* Click on the button labeled 'Next'.

Each time you click the 'Next' button, the next record in the database will be displayed.

## Data query puzzle

To see an example of a data query puzzle:

- \* Load the 'Profile' puzzle.
- \* Move the pointer to the camera icon, hold the shift key down and click the button
- \* Select Set stop mark from the Puzzle menu.

A stop mark should now appear on the camera icon in the puzzle. (The stop mark corresponds to the break point of a traditional programming language.)

- \* Solve the puzzle

When Show and Tell gets to the camera icon it will stop and shade the icon.

- \* Hold the shift key down and double click on the camera icon
- \* Select Continue from the Puzzle menu.

A window labeled 'Peek' appears which allows you to see Show and Tell search through the database for Darren's record. When Show and Tell is done searching, click in the close box of the peek window. Darren's picture now appears in the box marked 'Profile'. Using the stop mark as we did here, allows the execution of subroutines to be monitored.

## OIL INVESTMENT

The previous example introduced some basic mechanics of working with Show and Tell's database. The next example illustrates the possibilities of making a complete relational database system.

Load the "gas-pump" icon. The schema for our Oil portfolio is now displayed. We will use this database as the basis for several puzzles which extract information from the various records.

\* Select **Find** from the **Answer** menu.

The record for Gulf oil should now be displayed on the screen. Since the value of the stock is in dollars and cents, we may want to change the precision of the puzzle so that only two decimal places are displayed. To do this:

\* Select **Precision 2** from the **Goodies** menu.

The value of Gulf stock should now be 49.62. If you would like to thumb through the database you may do so by either activating the buttons or selecting find next from the answer menu.

The second puzzle in the drawer calculates the portfolio value of the Oil stocks. It does this by using the iteration box to look at each record in the oil database, calculate the value of the stock we own in that company, and add this to a running total. (This is all explained more fully in the User's Manual).

When you solve the puzzle, you will be able to see Show and Tell as it looks at each record in the database. We can also add a box to the puzzle that will let us see the running total accumulate.

- \* Reset the puzzle by selecting **Reset** from the **Puzzle** menu
- \* Select the box drawing icon.
- \* Draw a box (as in MacPaint) on top of the arrow leaving the iteration box.
- \* Solve the puzzle again.

This time the running total should appear in the box you just drew.

The "Big Oil" puzzle demonstrates one of the logical operators that Show and Tell provides. This puzzle will find oil companies for which we either own more than 200 shares, or whose current value is more than \$50. To find the first such oil company, select the Find entry under the Answer menu. To find each additional big oil company, select Find Next under the Answer menu.

The "Big \$" puzzle searches through the database, calculates the value of the stock and displays the names of all the companies in which we own more than \$5000 worth of stock.



## § 2.1 Editing in Show and Tell



Figure 2.1

### Creating Text and Graphics



The text tool is used for typing text and numbers.

- \* Select the tool by clicking on its icon.
- \* Move the pointer to the place you wish to enter text.
- \* Click the mouse button.
- \* Type the text.

Text may be entered either in a box, in which case the text is the value of the box, or in the background or name areas of the puzzle, in which case the text becomes part of the background or name of the puzzle.



The pencil is used for freehand drawing.

- \* Select the pencil by clicking on its icon.
- \* Move the pointer to the area you wish to draw on.
- \* Hold the button down as you move the mouse.

The pencil may only be used in the background or the name area.

The Camera is used to paste images and graphics into the name area or the background of a puzzle (see Chapter Four for use of the camera)

The Eraser is used to erase portions of the name area and background of a puzzle. (See Chapter Four)

Editing is the process of creating or modifying the box structure of a puzzle. The box structure is the set of boxes and arrows that appears in the editing window. In this section we will take a close look at the mechanics of creating and changing Show and Tell puzzles. We will start by looking at the constructive aspects of editing, how to draw boxes and arrows. Then we will look at the techniques and tools for changing preexistent puzzles. This will include moving, sizing, and deleting preexisting puzzle elements.

### The editing tools

Painting, woodworking, gardening, and just about every other creative activity have their own set of tools designed to perform the specific jobs that are unique to each activity. Creating and editing puzzles are also activities that require special tools. Show and Tell provides tools that are designed to make it easy to draw, move, reshape, or erase boxes and arrows.

The icons along the left side of the screen represent most of the tools you will need for editing puzzles. This group of icons is called the **Editing bar** (figure 2.1). To use an editing tool simply click on its icon. Each icon in figure 2.1 is just one of several tools in a family of related tools. To select a tool underneath an icon in the editing bar:

- \* Move the pointer to the portion of the black bar just to the right of the icon hiding it.
- \* Hold the mouse button down. A sidebar appears displaying all of the tools in the family.
- \* Drag the pointer to the desired tool.
- \* Release the button.

## Drawing boxes and arrows

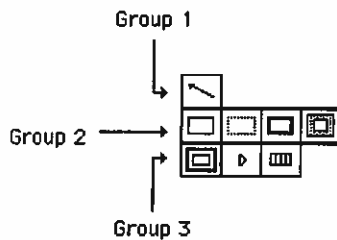


Figure 2.2

The tools for drawing boxes and arrows, grouped according to their usage, are shown in figure 2.2. The single tool in the first group is used to draw arrows. Each tool in group 2 is used to draw a box of the type shown in the icon. The tools in group 3 are used to draw a special type of box called the iteration box.

### Group 1

The arrow tool is used to draw arrows.  
to use it:

- \* Select the tool by clicking on its icon.
- \* Place the pointer at the point you wish the arrow to begin and click.
- \* Move the pointer to the point at which you wish the arrow to end and double click.

You may control the path the arrow follows by clicking, as you draw the arrow, at each place the path of the arrow makes a turn.

There are two restrictions on drawing arrows. Only arrows beginning and ending in boxes are allowed. You may not create a cycle. (A cycle is a path of boxes and arrows such that you can start out in a box, follow some arrows, and then end up back in the box you started from.)

### Group 2

These icons are used to draw boxes. They are used the same way the box tool in MacPaint is used.

To draw a box:

- \* Select the tool you want by clicking on its icon.
- \* Move the pointer to the place you want the top left corner of the box to be.
- \* Drag the pointer until the box is the desired size.

There are two restrictions on drawing boxes. A box must be at least one quarter of an inch long on each side, and no box may overlap another box (figure 2.3). If you draw a box that violates either of these conditions, the chime will sound and the box you drew will disappear.



Figure 2.3

## Group 3

The iteration box is unique in that it is not complete until one or more serial or parallel ports are added to it (ports are discussed in chapter three). We must first draw the iteration box and then add the serial and parallel ports.

\* Draw the iteration box in the manner described for group 2 boxes.

To draw a serial port:

- \* Select the serial port tool by clicking on its icon.
- \* Place the pointer on the edge of the iteration box at the point you wish the serial point to enter the box.
- \* Click the mouse button.

Note that for each serial port you place entering the iteration box, another serial port is placed leaving the iteration box.

To draw a parallel port:

- \* Select the parallel port tool by clicking on its icon.
- \* Place the pointer at the position you wish the port to be.
- \* Drag the pointer until the port is the desired size.

To use a port:

\* Draw an arrow that passes through the port. (Note: each port may have only one arrow passing through it.)

Often when drawing the box structure of a puzzle, mistakes are made and we need to correct them. Sometimes we want to change a puzzle to make it do something else or to add a feature. To do these things we need to be able to move objects and get rid of old ones. The next next two sections show how to do these operations.

The tools used to cut boxes and arrows are shown in figure 2.4.



Figure 2.4

Box scissors

To cut a box from the box structure of a puzzle:

- \* Select the tool by clicking on the icon.
- \* Move the pointer inside of the box you wish to cut and click. The smallest box containing the pointer will be cut.

Once the scissors have been selected, you may cut as many boxes as you wish without re-selecting the scissors each time you want to cut another box.

## Changing boxes and arrows

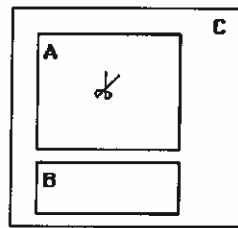
## Cutting boxes and arrows



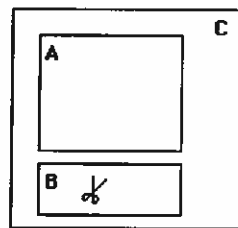


**Multiple box scissors.**

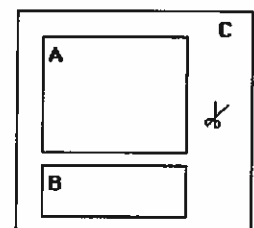
This tool is used to cut a box and all boxes that it contains. Figure 2.5 shows the relation between the position of the pointer and which boxes will be cut with the multiple box scissors.



Box A will be cut



Box B will be cut



Boxes A, B, and C will be cut

Figure 2.5



**Arrow scissors. To cut an arrow:**

- \* Select the tool by clicking on its icon.
- \* Place the pointer on or near the arrow you wish to delete and click the button.

**Moving boxes and arrows**

Show and Tell provides tools for moving boxes and arrows and a tool for reshaping a box. These tools are shown below.



**Box moving tool. To move boxes:**

- \* Select the tool by clicking on its icon.
- \* Place the pointer in the box you want to move and drag it to the desired location.

Just as you are not allowed to draw a box that overlaps another box, you are not allowed to move a box so that it overlaps another box. If you move a box that contains other boxes, the boxes inside the one you move will also move.



**Arrow moving tool. To move an arrow:**

- \* Select the tool by clicking on its icon.
- \* Place the pointer on the arrow you wish to move.
- \* Drag the arrow to the new position.

**Changing the size of a box**



It is also possible to change the size of a previously drawn box.

- \* Select the tool by clicking on its icon.
- \* Place the pointer on the lower right hand corner of the box.
- \* Drag the corner until the box is the desired shape and size.

**Iteration boxes**

Iteration boxes should not be moved nor expanded. Instead, cut them and redraw them.

## § 2.2 File Handling in Show and Tell

This section will describe how to save puzzles that have been created, and how to use puzzles that have already been saved. Puzzles are kept in drawers when they are not in use. To use a puzzle, move it out of the drawer. To save a puzzle, put it in a drawer. To throw a puzzle away, put it in the trash can. This is really all there is to file handling in Show and Tell.

### § 2.2.1 Drawers

#### Creating a new drawer

To create a new drawer:

- \* Select **New** under the **Drawers** menu.
- \* Type the name of the drawer in the dialogue window.
- \* Click on the **Save** button.

#### Opening a drawer

To open a drawer:

- \* Select **Open** under the **Drawers** menu.
- \* Double click on the name of the drawer you wish to open.
- \* If the drawer you wish to open is on another disk; close any drawers that are open, click on the eject button, remove the old disk, and insert the correct disk.

The drawer you opened should appear in the lower part of the screen. The **Arithmetic**, **Input Output**, and **Miscellaneous** drawers are also opened in this way.

#### Closing a drawer

To close a drawer, simply click in the close box in the top left corner of the drawer.

**Note** : A drawer may hold at most ten puzzles.

### § 2.2.2 Puzzles

#### Moving a puzzle

To move a puzzle from one drawer to another drawer:

- \* Open both the drawer containing the puzzle you want to move and the drawer you wish to put the puzzle in.
- \* Drag the puzzle from its current drawer into the new drawer.

## Saving a puzzle

Puzzles are saved in drawers. To save a puzzle (except a recursive puzzle):

- \* Open the drawer you wish to save the puzzle in.
- \* Place the pointer in the current puzzle area.
- \* Drag the icon from the current puzzle area onto a blank portion of the drawer and then release the mouse button.

Only the currently displayed puzzle may be saved. If you have made changes to a previously saved puzzle and wish to save the changes, drag the icon from the current puzzle area onto the icon of the puzzle in the drawer. All puzzles called as subroutines from other puzzles should be kept on the same disk as the puzzle that calls them.

## Saving a recursive puzzle

Recursive puzzles are puzzles that call themselves as subroutines. Due to their special nature, recursive puzzles must be saved in a special manner.

- \* Draw the box structure of the puzzle but leave any box that will contain a recursive call blank (leave all boxes that will eventually contain the name of the puzzle blank).
- \* Save the puzzle
- \* For each occurrence of the recursive call, drag the name of the puzzle from the drawer into the correct box in the editing window.
- \* After all boxes are filled in, save the puzzle again by dragging the icon from the current puzzle area onto the puzzle icon in the drawer (see figures 2.7-2.8).

## Saving changes to a recursive puzzle

- \* Save the puzzle that is changed.
- \* For each occurrence of the recursive call, drag the name of the puzzle from the drawer into the correct box in the editing window.
- \* After all boxes are filled in, save the puzzle again by dragging the icon from the current puzzle area onto the puzzle icon in the drawer.

## Opening a puzzle

In order to use a puzzle, it must be placed on the desktop. To do this we must move the puzzle out of the drawer.

- \* Open the drawer containing the desired puzzle.
- \* Place the pointer on the puzzle's icon.
- \* Drag the icon into the current puzzle area and release the mouse button.

Another method of opening a puzzle:

- \* Double click on the puzzle icon in a drawer.

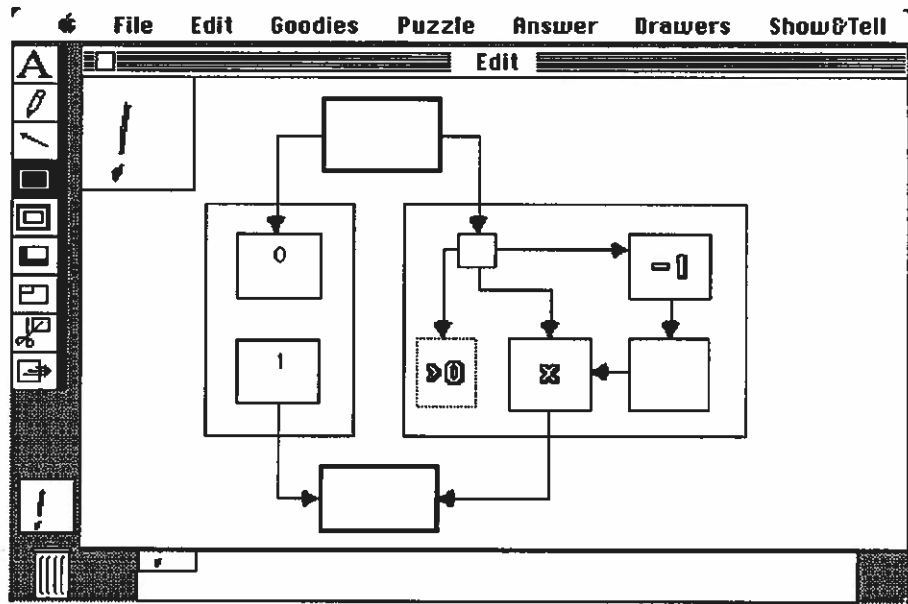


Figure 2.7

The first step in making a recursive puzzle is to make the box structure, but without the recursive call. This incomplete puzzle is then saved (figure 2.7).

The next step is to put in the recursive calls (figure 2.8). Finally, the puzzle is saved in its completed form.

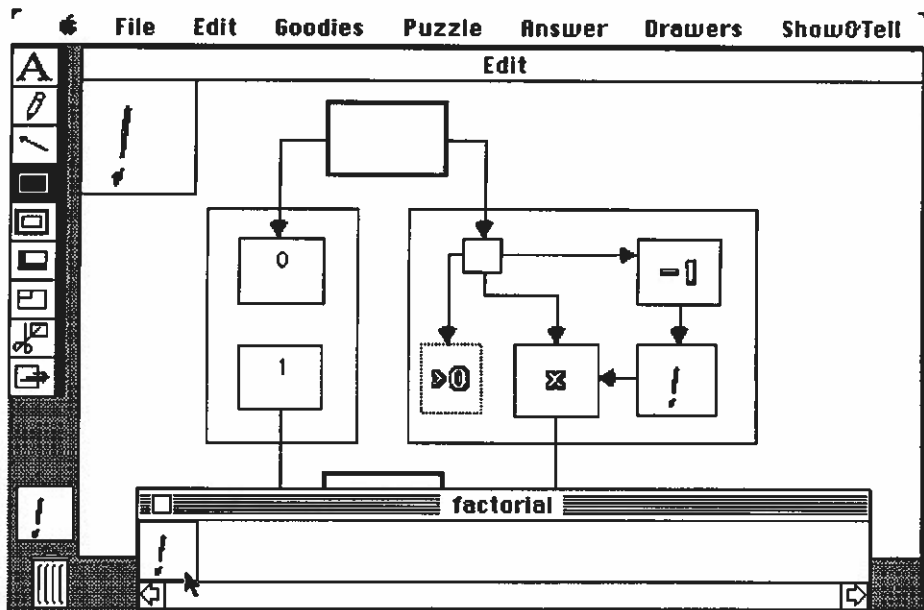


Figure 2.8

## § 2.3 Database

This section describes procedures for creating, changing, and working with database puzzles. If the notion of a database is unfamiliar to you, read the section on databases in the next chapter "Programming in Show and Tell."

### § 2.3.1 The schema

The first step in creating a database puzzle is planning the layout of the database. You will need to know what information you want to store in the database and how you want it arranged on the paper. Once you have the *schema* planned, you can create the puzzle.

#### Creating the database schema

- \* Select **New** from the **File** menu.
- \* Select the base box drawing tool from the editing bar.
- \* For each field in the database, draw a *base* box of appropriate size.
- \* If you wish to label the boxes you may do so now by putting labels in the background. (Use the text and graphics tools)
- \* Name the database by drawing a name in the Name area.
- \* Save the puzzle in a drawer.

### § 2.3.2 Database solutions

We now have the schema or record structure of the puzzle stored in the computer. We can begin to enter the solutions to the database. Enter each solution to the database as follows:

#### Entering solutions into the database

- \* Select **Buttons** from the **Answer** menu.
- \* Type the information for one solution (record) into the appropriate base boxes.
- \* Click in the **Save** button to save the solution.

Sometimes we wish to change the information in the database. For instance, someone may have a new phone number and we would like to replace the old phone number with the new one. Sometimes we just want to get rid of a record.

#### Destroying solutions

To destroy a record:

- \* Make sure the record you want destroyed is currently displayed.
- \* If the buttons are not currently displayed, select **Buttons** from the **Answer** menu.
- \* Click in the **Destroy** button.

Be sure that you want to destroy the record. Once it is gone you can not retrieve it.

#### Changing solutions

To change a solution in a database (e.g. change someone's phone number):

- \* Display the solution that is to be changed (e.g. use the **next** or **previous** buttons to display the solution).
- \* Edit the solution until the correct information is displayed.
- \* Click on the **Change** button.



## Data query

There are two forms of data query available in Show and Tell. The first form is suitable for simple queries that are not performed often. The second method is suitable for complicated queries, especially ones involving calculation, and for queries that are performed often.

Simple data queries can be accomplished by clearing the base boxes and then typing in a pattern to match. Show and Tell will then display the first solution that is consistent with the given information. This type of query was demonstrated in the first chapter (p.10) when Jane's picture was found.

Complex queries may be put in the form of a puzzle that takes the database as its input. This type of query was demonstrated by the oil investment examples in chapter one (pp. 10ff).

## Chapter Three: Programming in Show and Tell

### Computer programs

This chapter introduces the programming philosophy behind Show and Tell. It is designed for people who are new to programming or who are new to the Show and Tell environment. The concepts of puzzle, completion, solution, and consistency are all introduced. Two classes of computer problems are introduced and the method Show and Tell uses to solve each class is discussed.

Computer programs do two jobs. They serve as a medium for communication between computers and the people who use them, and they serve as a language for rigorous definition of problems. Show and Tell differs from traditional computer languages by being a visual language. Most programs are written in words but Show and Tell uses boxes and arrows to define the structure of a program.

We can think of problems as if they were puzzles with a piece missing. For example, " $2+3= \square$ " is a kind of puzzle which has a missing piece. It is up to us to find the missing piece and put it in the puzzle. "5" is the missing piece to the puzzle we just saw. We will call the missing piece(s) to a puzzle its **solution**. The process of finding the solution is called **completion** because we complete the puzzle by finding the missing piece(s).

A puzzle is the means that Show and Tell provides for asking the computer for information. There are only two ways the computer can get a solution for the puzzle. It can either try to *generate* a solution for the problem, or it can *look up the answer* in a table of information. For instance, to find the square of a number, you might give Show and Tell a puzzle which tells the computer that the square of a number is that number multiplied by itself. When you ask the computer to find the square of a particular number, it will know how to find the answer. This is an example of generating the solution to a puzzle (computation). If the computer had a telephone book stored in it, then you might ask it to find John's phone number. The computer would then look through the phone book until it found John's name. When it had found his name, the computer would then be able to tell you John's phone number. This is an example of looking up the answer (database).

### Puzzle criteria

Not every combination of boxes and arrows is a meaningful puzzle. There are certain requirements that must be met before Show and Tell will recognize a set of boxes and arrows as a puzzle. There are three features common to all puzzles:

1. The puzzle definition is consistent (it makes sense).
2. The definition of the problem is complete and precise.
3. There is a missing piece, a need for completion.

The first characteristic of puzzles is their consistency. If you want Show and Tell to find a solution for you, you have to make a puzzle that has a solution. You can draw some boxes and arrows that say "find a negative number greater than zero," but this doesn't make any sense since there is no negative number that is greater than zero. There is something inconsistent about the definition. Later we will learn how inconsistency can be used to help define a problem, but there can be no solution to a puzzle whose definition is always inconsistent.

**How a puzzle defines a problem**

The second criterion requires that we be precise and complete in defining the puzzle. We want the solutions for the puzzle to be all and only those solutions we are looking for. Unlike a magician, Show and Tell cannot guess what number you are thinking about. You have to describe the solution to the problem clearly enough so that when Show and Tell comes across the answer, it will be able to recognize it.

Finally, there has to be a blank to fill in. If you do not leave a blank box then either you have figured out the problem yourself or you have not defined it properly. On the other hand, just leaving a blank box in the puzzle someplace won't do either. You have to put the empty box at the right place in the puzzle.

There are two methods by which Show and Tell comes up with a solution to a puzzle. The first is by trying to construct the solution. This involves looking at the given information and trying to **make** a solution that fits the description. The second method is to try and **find** the defined solution in a list that already exists. The first method is called **computation** and the second method is called **database**. The next two sections introduce the fundamental ideas of each method.

**Defining computation problems**

A puzzle is a set of boxes and arrows. The puzzle defines a problem by putting things in boxes that may be connected with arrows. Boxes hold values that may be numbers, pictures, words, or even other puzzles. Whatever is in a box is called the value of that box. The simplest box-structure is an empty box. This is not a very useful puzzle since neither you nor Show and Tell knows what to put into it. It fails to satisfy our second condition. Somehow we need to explain what sort of things we want Show and Tell to put into the box, that is we need to define what a solution to the problem is or looks like.

Suppose our problem is to find the square of 335. The number is big enough that we would rather have the computer do the problem for us. We know that we need more than just the empty box in our puzzle. Although we don't know what the square of 335 is, we do know the definition of what a square is in the general case. The square of a number is that number multiplied by itself. If we can somehow translate this English definition into the language that Show and Tell recognizes, we would be close to finding our answer. If Show and Tell knows what a square of a number is, then we can ask it to find the square of a particular number, e.g. 335.

Right now, Show and Tell does not know what the square of a number is. But if we look at our definition of square, we see that if we know how to multiply then we can find the square of any number. Show and Tell does know how to multiply. There is a definition, a puzzle, already made up that tells Show and Tell how to find the product of two numbers. The multiplication primitive and many others are found in the arithmetic drawer.

Now we need a way to indicate that we want to multiply two numbers and put the answer in the empty box. To do this, Show and Tell lets one box control what value goes in another box. This is done with an arrow.

There are two puzzles in figure 3.1. Both of them contain two boxes and one arrow. The two boxes labeled A. are a puzzle. The box containing "3" has an arrow pointing to the empty box. The arrow means that the first box controls the value of the second box. Another way of saying this is that an arrow imposes constraints upon the values that may go into the box that the arrow points to. The constraints that an arrow imposes are determined by the the box from which the arrow begins.

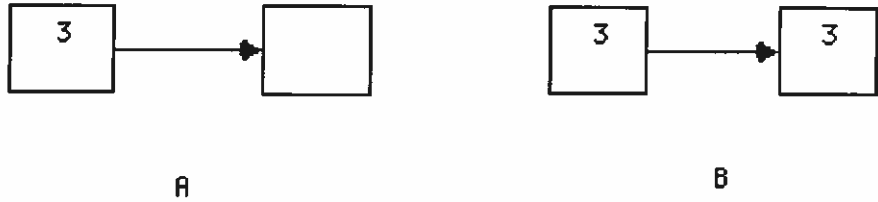


Figure 3.1

In our example the first box tells the second box that its value should be "3". If we tell the computer to find a solution to the puzzle, it will give us the puzzle shown in B. By filling the second box in with "3" Show and Tell has completed the puzzle. All the boxes have something inside of them. Furthermore, the puzzle is now **consistent**. This means that each box has the value that it is told to have. The first box has no constraints imposed upon it so its value may be anything. The second box is being told that its value should be 3. Indeed the value of the second box is 3 so it is well behaved. Since all of the boxes in the puzzle have a value that is consistent with constraints imposed upon them, the puzzle as a whole is consistent.

We now return to the problem of finding a puzzle that will generate the square of a number. Since we can tell a box what its value should be, we just need to tell an empty box to be the product of two numbers. If we make the numbers the same, then the puzzle will tell the empty box to be the square of that number.

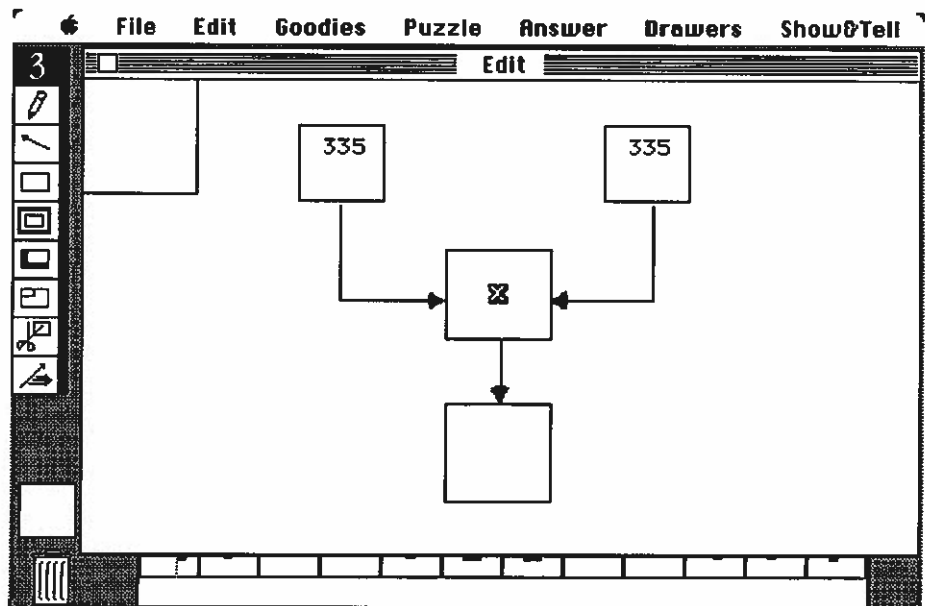


Figure 3.2

The puzzle in figure 3.2 will find the square of 335. The empty box will get the value of the multiplication box (the one with an "x" in it). This is because of the arrow. The "x" is the symbol that Show and Tell uses to mean "multiplication". The two boxes with 335 in them are telling the multiplication box what numbers to multiply together. The completed puzzle is shown in figure 3.3.

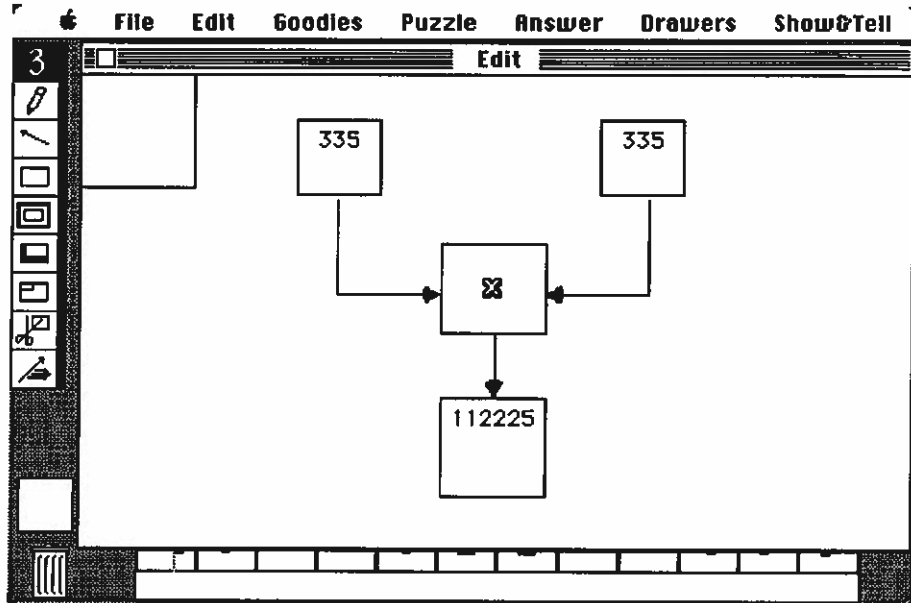


Figure 3.3

## Iteration

This section of the manual is an introduction to the idea of iteration and the use of the iteration box. Iteration simply means repetition. The box structure in the iteration box is copied several times before the puzzle is solved. This section will explain exactly how the copying is done and give examples of iteration.

There are two types of iteration in Show and Tell, **serial iteration** and **parallel iteration**. Serial iteration is introduced first followed by parallel iteration. Finally, the combined use of serial iteration with parallel iteration is introduced.

## Serial iteration

Serial iteration is indicated by the presence of serial ports on the iteration box. A serial port is a small triangle touching the edge of the box. The simplest example of serial iteration is the counting problem. To illustrate serial iteration we will make a puzzle that counts numbers starting from one (see figure 3.4). The box made out of double lines is the iteration box. The two triangles connected to it are called the serial ports. The triangle with the point touching the iteration box is the port which lets information into the iteration box. The triangle with the side touching the edge of the iteration box is the port through which information leaves the iteration box.

When this puzzle is solved, it will start counting from one and keep on going until we tell it to stop. The numbers will appear in the

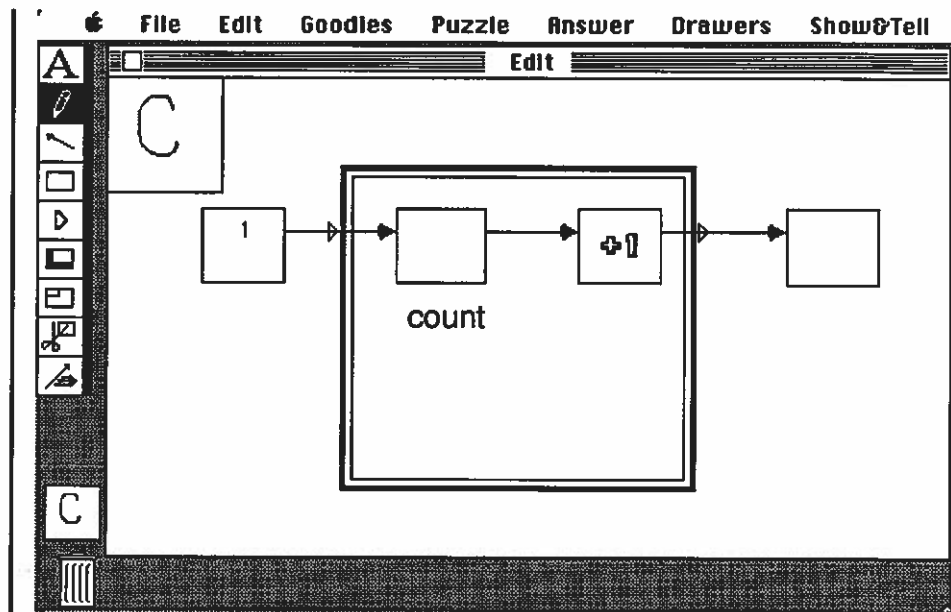


Figure 3.4

empty box labeled "count". The iteration box works in a very simple way. It solves the part of the puzzle that is inside the box. Then it takes whatever is inside of it, copies it and then hooks the output of the original to the input of the copy. An illustration of how this works for the counting problem is shown in figure 3.5. The three dots indicate that this process continues forever (or until we tell the computer to stop).

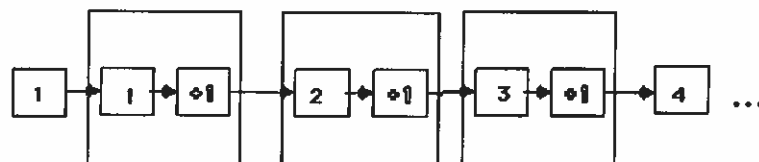


Figure 3.5

The iteration box is just a convenient way of *folding* all of the big boxes into one box. The iteration box may be thought of as acting like a paper doll. When you make paper dolls, you start with a folded piece of paper and then cut the outline of the doll once. When you unfold the paper, you have a string of dolls, all alike, connected at the hands and feet. The same is true of the iteration box. The iteration box is like the sheet of folded paper. You can draw a puzzle in the box and when you solve it, many copies of the puzzle are made and each copy is connected to the next by the serial ports.

In real applications we want the iteration to stop. If it does not stop then there will be no final answer. Serial iteration stops only if one of the copies of the iteration box becomes *inconsistent*. The counting puzzle we made can never become inconsistent so it will never stop. If we want to make the counting stop, we need to add a test to the box structure.

Suppose we want to count from one up to ten. The puzzle in figure 3.6 does this. The additional boxes tell the computer when it has reached ten. The iteration box will continue making copies of the puzzle until there is an inconsistent copy. When the inconsistency arises, the previous solution to the puzzle is returned as the *value* for the iteration box.

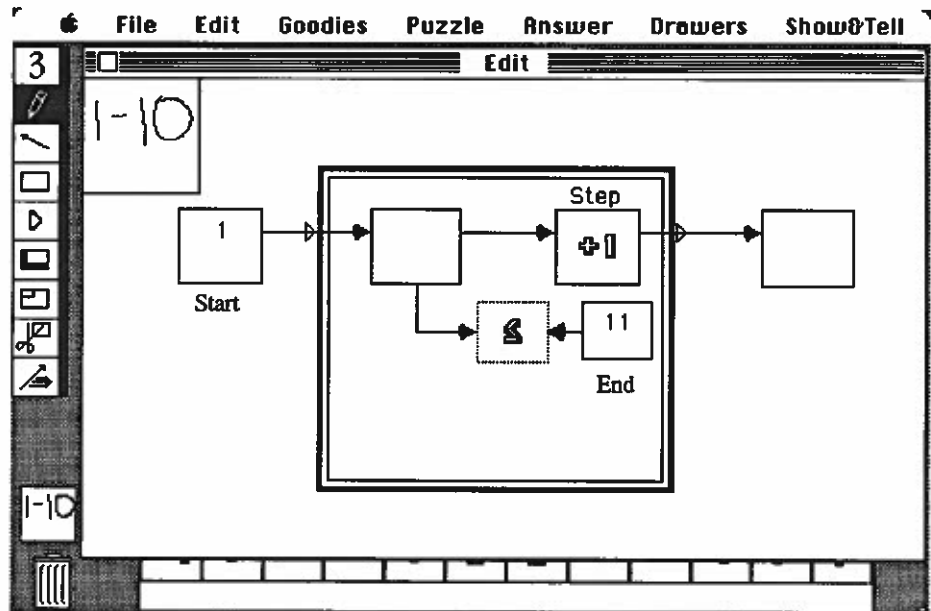


Figure 3.6

The counting problems we have been looking at have dealt with generating sequences of numbers. A sequence is just a list of numbers. For example "1,2,3,4,5,..." is the first sequence that we defined. Other sequences are: "0, 5, 10, 15", "2,3,5,7,11,..." etc.

There are three things that we need to know in order to define a sequence. We need to know where to start, where to stop, and, if given a number (or numbers) in the sequence, how to find the next number. If we have these three pieces of information then we can make a Show and Tell puzzle to generate the sequence. The puzzle we have can be modified to generate a large variety of sequences. To modify this puzzle to generate a particular sequence, put the starting value in the box labeled START, put the ending value in the box labeled END, and put the definition of how to figure out the next number in the box labeled STEP.

To illustrate the versatility of the puzzle we can make it count down from twenty to four by two's (20, 18, 16, 14, ..., 4, 2). To do this we simply fill in the blanks in our schema. We want to start at twenty so we put "20" in the START box. We want to end at two so we put "2" in the END box. Since we subtract two from the current number to get the next number, we put a puzzle that subtracts two from a given number into the STEP box. The puzzle is shown in figure 3.7 (the "-2" icon is a puzzle that we made that subtracts two from whatever number we give it).

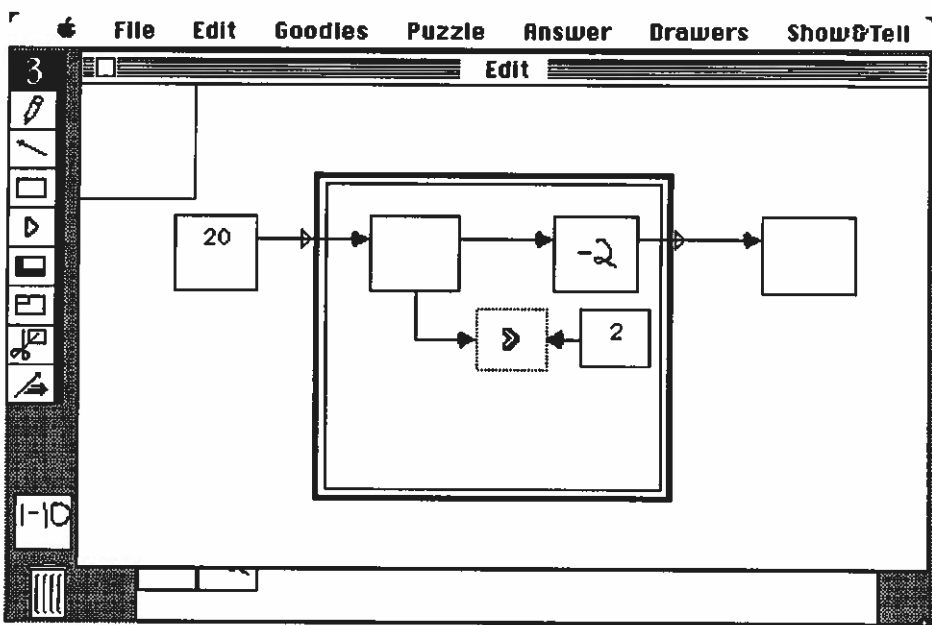


Figure 3.7

In this section we will use what we learned about sequences to build more useful and more complex puzzles. One problem that often comes up in math and science is that of finding the sum of a sequence of numbers.

**The summation example**

Suppose the problem is to find the sum of the numbers from one to ten. Our plan is to use the sequence generator we made in the last section to make the numbers from one to ten. We will then add a puzzle that keeps track of the running total. A puzzle that does this is shown in figure 3.8.

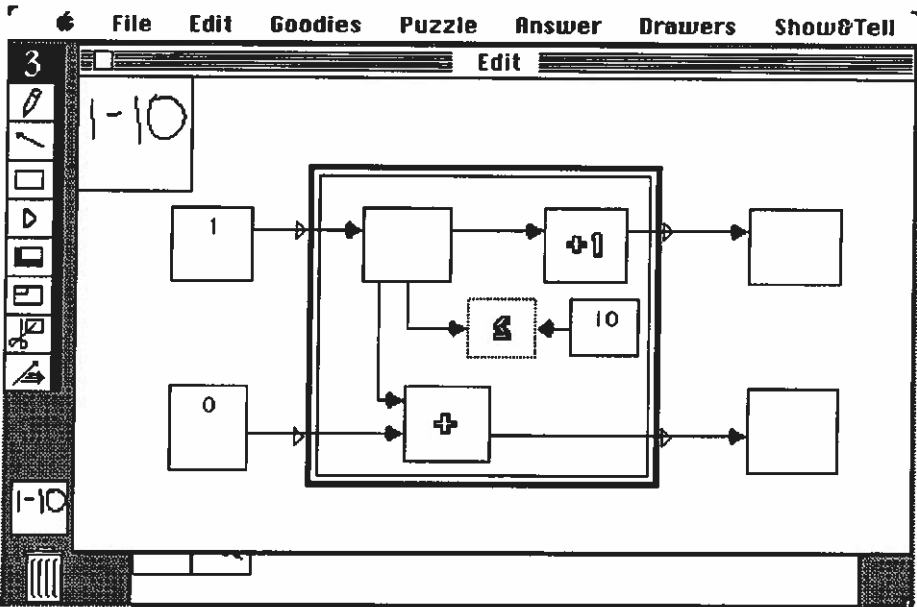


Figure 3.8



Parallel iteration

The top row of boxes should be familiar to you. That part of the box-structure is just copied from the puzzle to generate the numbers from one to ten that we saw in the last section. The lower part of the puzzle is the part that takes care of the running total. The idea is that each time we generate a new number in the sequence, we add it to the running total. When the sequence is done generating, we know that we have added all of the numbers in the sequence. The way we get each number is to "tap" the stream of numbers going across the top. The blank box at the top of the screen takes a copy of each number in the sequence and sends it to the bottom part of the puzzle.

We have seen one way of using the iteration box, serial iteration. There is another way we can use the iteration box and this is known as **parallel iteration**. Parallel iteration is indicated by the presence of a parallel port on an edge of the iteration box.

Serial iteration continually refined one set of values until it was the desired output. There was one input set and one output set. Parallel iteration is used when you want to do the same thing to a several sets of input. There are many starting values in parallel iteration and for each starting value a copy of the puzzle is made. Each copy then solves the puzzle for its particular input value. There is thus as many solutions generated by parallel iteration as there are input values. The same folding that occurs in serial iteration is present in parallel iteration but the input values to each copy are provided in parallel.

Suppose we have a list of students and their grades in a file. They have all just taken a test and we want to add the test scores to their cumulative totals. Amazingly enough, all of the students recieved a 95 on the test. The problem is to add 95 to each of the numbers in a list. The puzzle to do this is shown in figure 3.9.

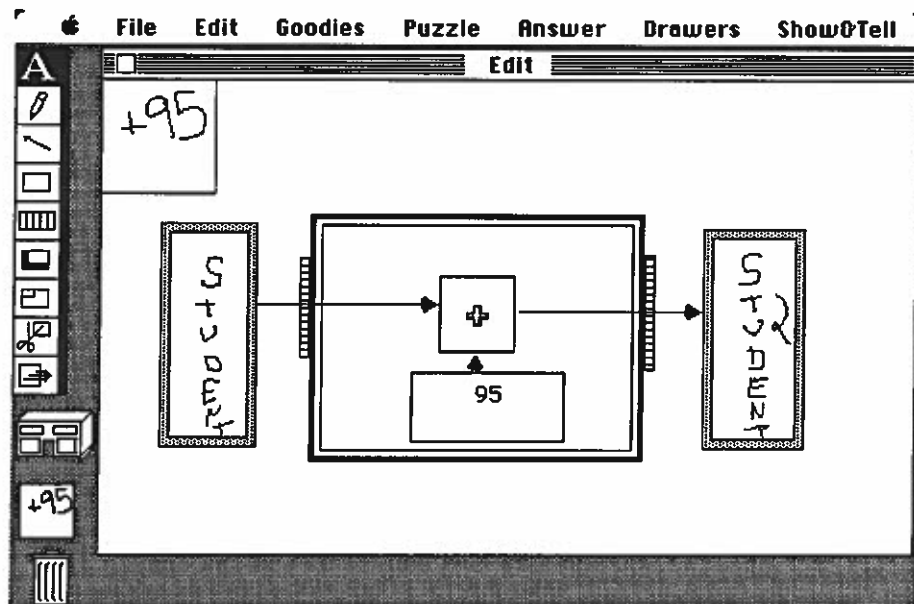


Figure 3.9

The box marked input file is a list of the students grades. The arrow going from the file box into the iteration box passes through a parallel port. In parallel iteration, like serial iteration, many copies of the iteration box are made. The difference between the two types of iteration is in how the copies are hooked together. In serial iteration, they were hooked together like paper dolls. When one copy was done solving its puzzle, it gave the solution to the next copy in line. In parallel iteration, everything is different.

Each item in the input file gets its own copy of the puzzle. And the output from a parallel iteration is not a single number but rather a number for each copy of the puzzle. The puzzle as it would look in the expanded version is shown in figure 3.10. (There are just three students in this example because we don't have room for any more on the paper.)

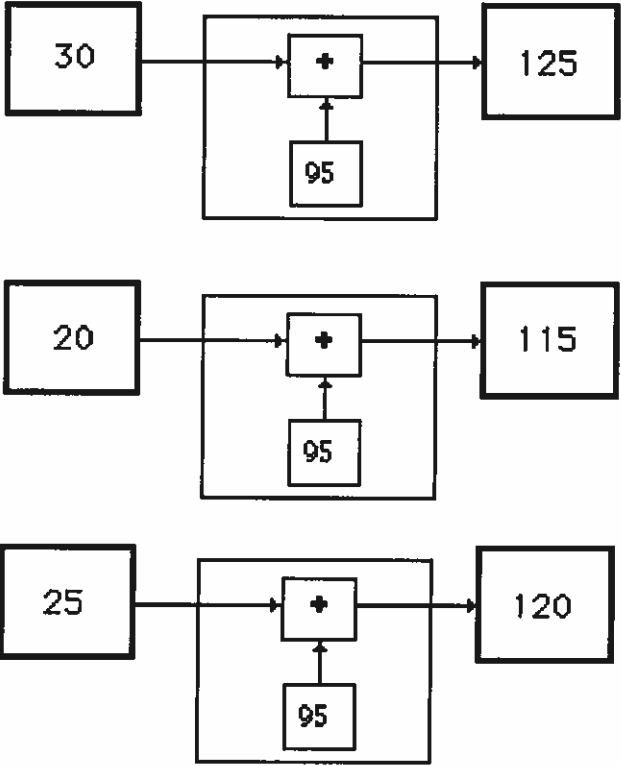


Figure 3.10

The puzzle in figure 3.11 finds the average grade for the students in CS455. The students names and grades are in the file "CS455" which enters the iteration box at the bottom through the parallel port. The file is fed into a structure box. The structure box shows us the fields in the database. We are only concerned with the grades and not the names. The serial ports on the left part of the iteration box keep a running total of all the grades of the class. The serial ports in the right hand portion of the iteration box keep track of how many students there are in the class. Finally, the total of the grades is divided by the number of students in the class and the answer is put into the box at the far left of the puzzle.



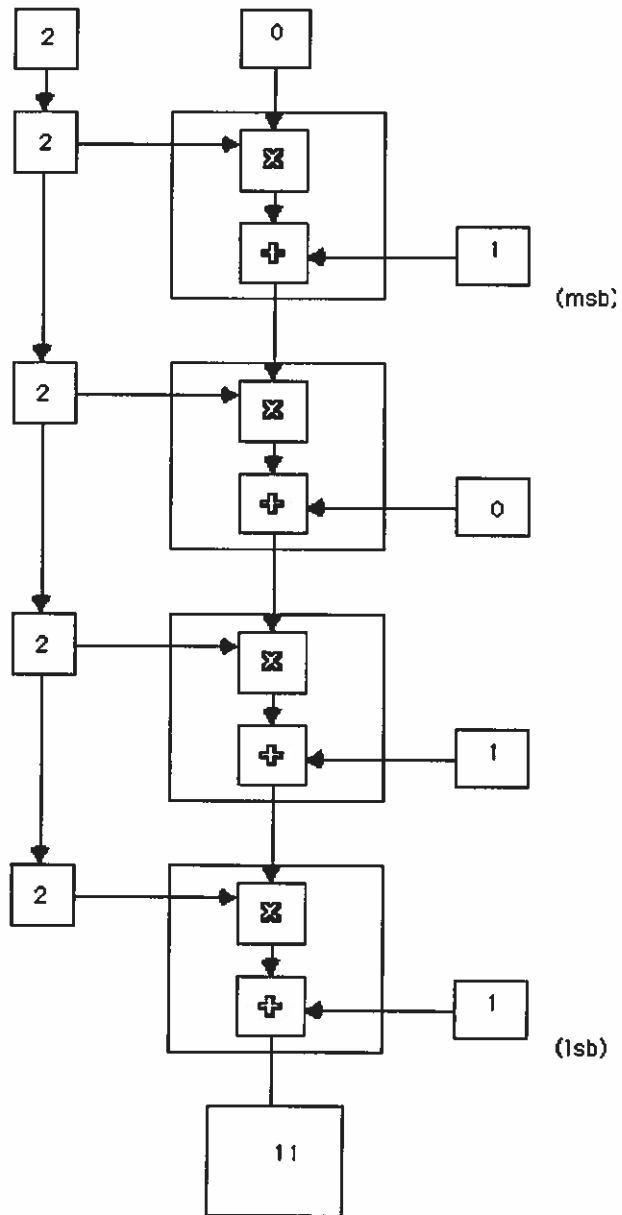


Figure 3.13

We have briefly seen how a puzzle defines a computation problem. Throughout the manual there will be more examples of Show and Tell puzzles. However dissimilar puzzles may seem, they are all based on the ideas of finding a solution to the puzzle that is consistent with the given constraints. More advanced techniques for defining problems will be explained in later chapters. For more precise and complete definitions of boxes, arrows, and puzzles, consult the reference section.

## Defining a database puzzle

There are many questions or problems that can not be solved by *computing* an answer. The solution to the problem is *found* in a table of information. If your problem is to determine how many people live in South Dakota, you have to look it up in an atlas or an almanac. There is no formula for computing a solution to the problem. If you want to know if John is in the third grade, you have to look at the list of third graders and see if his name is in the list. Problems like these in which you have to consult a list are called **database problems**.

## Solutions

A database differs from the puzzles that we have seen so far by having a set of solutions associated with the puzzle. The square puzzle had to generate the solution. In a database we have to look up the answer in a list of solutions. Each solution to a database is a set of one or more related values. A phone book is a good illustration. Each solution in the phone book has three values; the name of the person, the address, and the phone number. To use a database, you give the computer some information and the computer will supply the rest. For instance, when we use the phone book we provide the name of the person and the phone book tells us what the phone number and address for that person are. The combination of a person's name, address, and phone number is considered one solution to the database. To illustrate the idea of a database puzzle, we will look at a simple database designed to keep track of stocks. The puzzle is shown in figure 3.14.

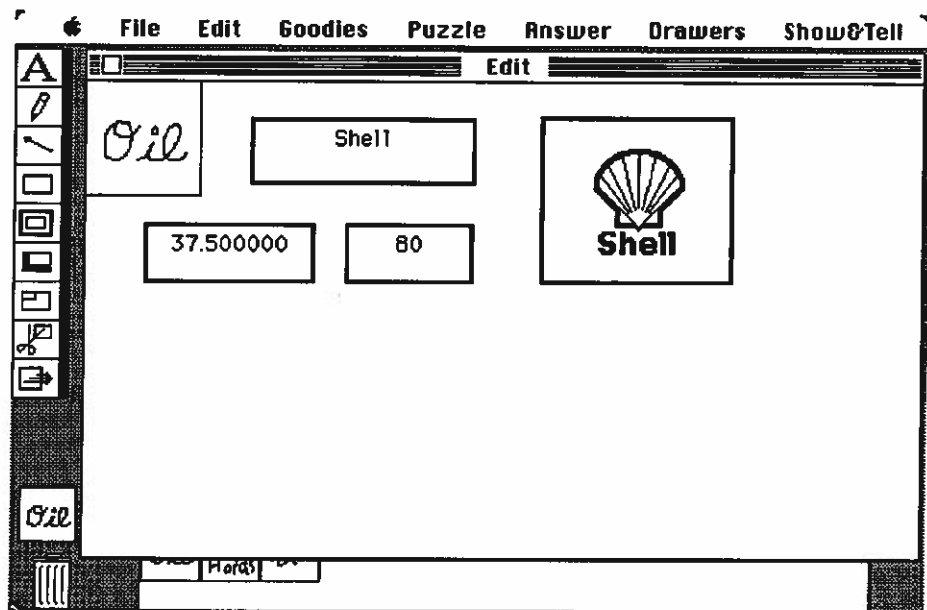


Figure 3.14

In this example the four base boxes represent the four pieces of information that we want for each of our stocks; the name of the company, the price of the stock, the number of shares we own, and a picture of the company logo. The box-structure of the database is the schema. It contains a base box for each piece of information we want to save. You may think of the database as a recipe box full of index cards. Each card in the recipe box has a piece of information that is the value of one of the boxes in the schema. In our example each company has its own card.

The basic idea in finding a solution to a database problem is the notion of pattern matching. We impose constraints on some of the base boxes in the schema and Show and Tell tries to find a solution (index card) that meets all of the constraints. Just like in the computation problems, constraints are imposed by arrows.

With the database we are only able to see one record (card) at a time. In figure 3.14 we currently see the card, or record, for Shell Oil. There are other records in the database. For example there is one for Mobil Oil. If we want to see the record for a specific company we can search through the database (using the buttons explained in section two). But this may take a long time if there are a large number of solutions. We can also make Show and Tell do the searching for us. To search for Mobil Oil, all we have to do is draw a box an arrow restricting the company name box (figure 3.15 ). Only solutions that have "Mobil Oil" in the company name field will be found. When it finds the card, it will display it in the box-structure. Now we can see how much we paid for Mobil stock and how many shares we own (figure 3.16).

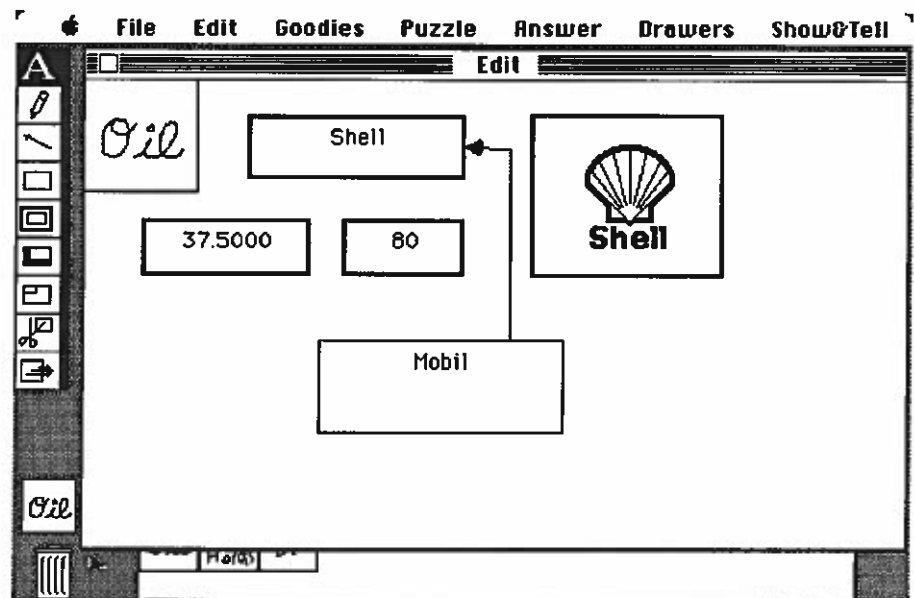


Figure 3.15

Another way of making the same query is to clear the base boxes, type in the name of the oil company, and then chose **Find** under the **Answer** menu.

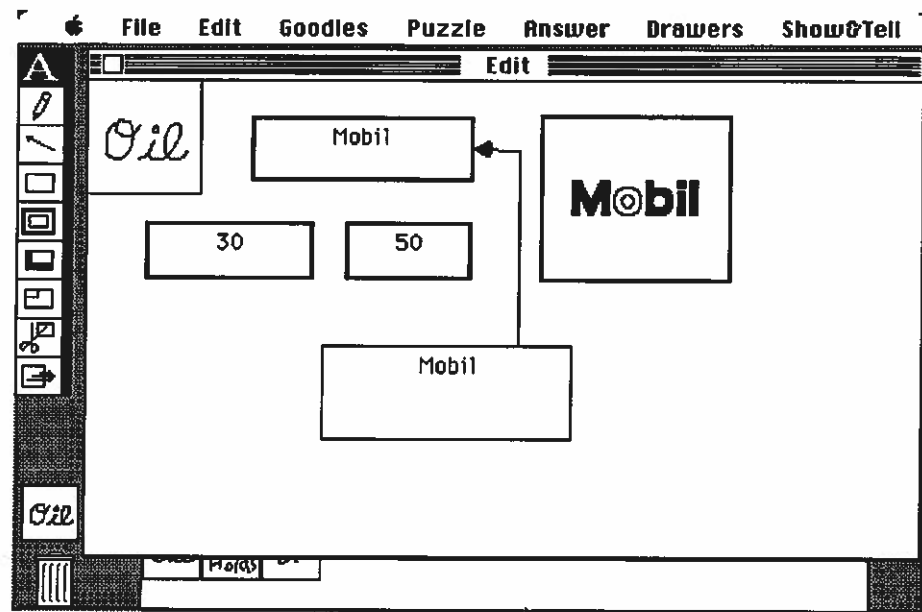


Figure 3.16

The set of database solutions may also be considered as a file and used as the input to an iteration box. The puzzle "Oil Holds" is an example of a database used in this capacity (figure 3.17). This puzzle will compute the total value of all the stocks in the "Oil" database. The idea of the puzzle is that the total value is the sum of the values of the individual stocks. Since the database does not have a subtotal for each stock, we will also have to figure that out.

The file "Oil" is feed into the iteration box through a parallel port and then into a structure box. The base boxes in the structure box represent the various fields in the "Oil" database. The puzzle multiplies the number of shares by the price per share and then adds this to the running total at the bottom. The iteration stops when all of the solutions in the "Oil" database have been considered.

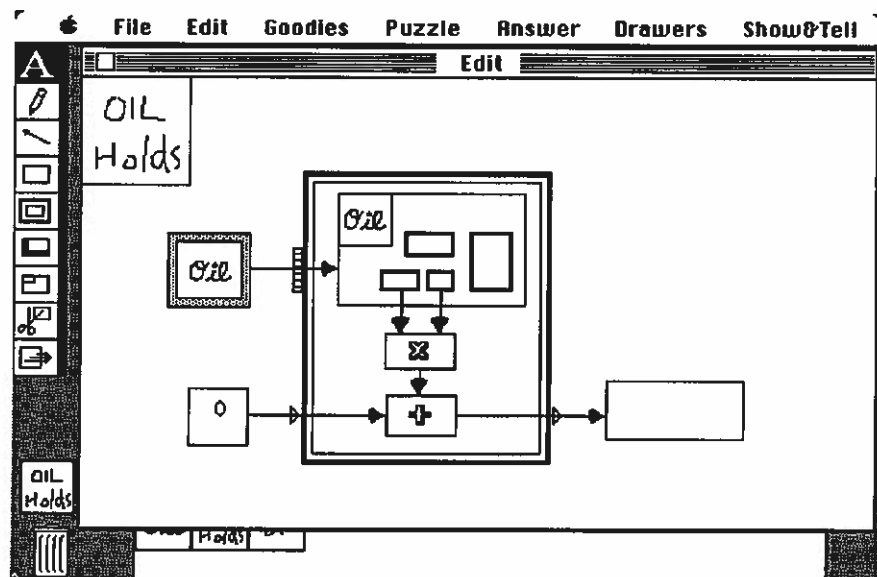


Figure 3.17

## § 4.1 The Show and Tell Environment

The parts of the Show and Tell screen are shown in figure 4.1. Each element is briefly described below. For more detailed information regarding a specific item see the entry in the reference section for that item.

### Menu Bar

The Menu Bar contains the Show and Tell menus. Each menu contains a group of related commands.

### Editing Icons

These icons represent the tools needed for editing a Show and Tell puzzle.

### Current Puzzle Area

The icon that appears in the current puzzle area determines which puzzle is in the currently active window. To save the current puzzle into a drawer, move the icon found here into the desired drawer. To select a new current puzzle, drag the icon of the puzzle from the drawer to this box and the new puzzle will become the currently displayed puzzle.

### Trash Can

The trash can is used for throwing away puzzles that are no longer needed. To throw away a puzzle, drag its icon from its drawer into the trash can. Note- throwing away a puzzle is an irreversible process. Once a puzzle has been thrown away, it no longer exists.

### Name Area

The name area holds the name of the puzzle. The name area may be edited using the pencil, eraser, camera, or typing. The name area's size may be changed using the box expansion tool (see page 42).

### Background

The background is used for holding comments about the puzzle. Nothing in the background will affect solutions to the puzzle. The background of the puzzle may be edited with the pencil, eraser, camera, or typing.

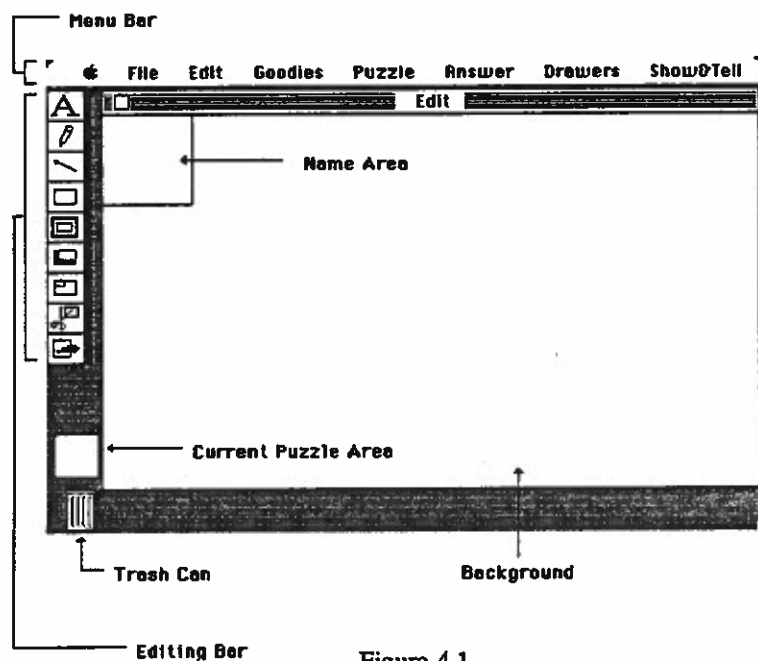


Figure 4.1



## §4.2 The Elements of Show and Tell

This section discusses all of the objects in Show and Tell.

### Puzzle

A puzzle consists of the following parts.

- \* Name of the puzzle
- \* Box-structure
- \* Set of known solutions for the puzzle (database)
- \* Background

### Arrows

Arrows are used to connect boxes to form the box-structure of a Show and Tell puzzle. An arrow connects one box with exactly one other box. A box may be connected to several other boxes, but each relationship is represented by a different arrow.

There are two restrictions on the interconnections of boxes in a box structure.

- \* An arrow may not connect a box to itself.
- \* There may not be a series of boxes and arrows that form a cycle. A cycle is a series of boxes and arrows such that you can start out in a box and by following the arrows you can get back to that box.

The Show and Tell editor will not allow you to draw an arrow that results in a box structure that has either of these conditions.

### Boxes

The box is the basic building block of Show and Tell. Boxes are connected by arrows to form box structures. Show and Tell finds solutions to box-structures when it completes the puzzle. The various boxes are discussed in this section.

### Box Structure

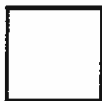
A box structure is a set of boxes connected by a set of arrows. Only boxes and arrows drawn with the box and arrow editing icons are considered part of the box structure. Boxes and arrows drawn with the pencil or imported from a Macpaint file are considered to be in the background.

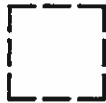
The objects that can be placed in a box may be as complex as an entire Show and Tell puzzle, or as simple as a number. The following objects may be placed in any Show and Tell box unless otherwise indicated:

- \* name of a puzzle (its icon)
- \* a box-structure
- \* data object
  - \* text
  - \* number
  - \* image

### Closed Box

The primary function of the closed box is to hold a Show and Tell object while containing any inconsistency within the box.





**Open Box**

The primary function of the open box is to hold an object while allowing any inconsistency that arises within the box to flow out into the surrounding environment.

**Base Box**

Base boxes serve two functions in Show and Tell; they indicate the in-boxes and out-boxes of a puzzle that is used by another puzzle, and they are used in a database puzzle to indicate what information comprises a solution to the database.

**In-box**

An in box is a base box that has only out going arrows connected to it (figure 4.2). The in box is used to indicate which boxes in the box structure are parameteres of the puzzle. These values may be typed in by the user, or passed to it by another puzzle.

**Out-box**

An out-box is a base box that has only incoming arrows connected to it (figure 4.2). The out-box is used to hold a value that is part of the solution to the puzzle. This value may then be passed to another puzzle.

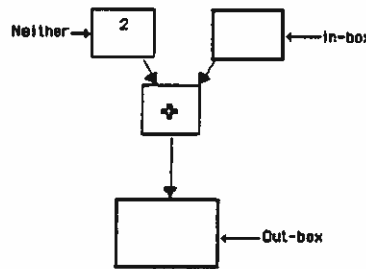


Figure 4.2



**Iteration Box**

The iteration box is used to indicate that the contents of the iteration box are to be solved several times. For an introduction to the notion of iteration, see chapter three of the manual.

There are three ways to make a connection to part of the box structure contained in an iteration box; through a serial port, through a parallel port, or through a direct connection from outside of the iteration box (figure 4.3).

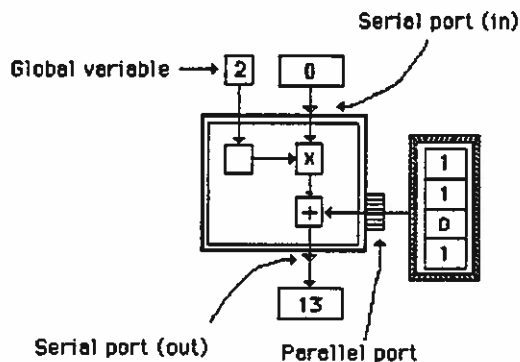
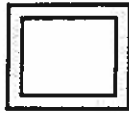


Figure 4.3



**File Box**

The only object that may be put into a file box is (the icon for) a file. A file is a sequence of solutions for a particular puzzle. The file box may be used as either an input to a box structure, via a parallel port, or as an output that collects a series of solutions to a puzzle.

**Structure Box**

Many times we use one puzzle solution as part of another. Often we want to connect certain parts of a solution to various parts of another solution. The structure box allows us to do this explicitly.

The structure box has a name area. The base boxes of whatever puzzle we drag into the name area of a structure box appear in the rest of the structure box. We can then connect these base boxes to the rest of the puzzle (figure 4.4). A structure box breaks apart a solution (record) of a database making each field of the solution accessible to the rest of the puzzle.

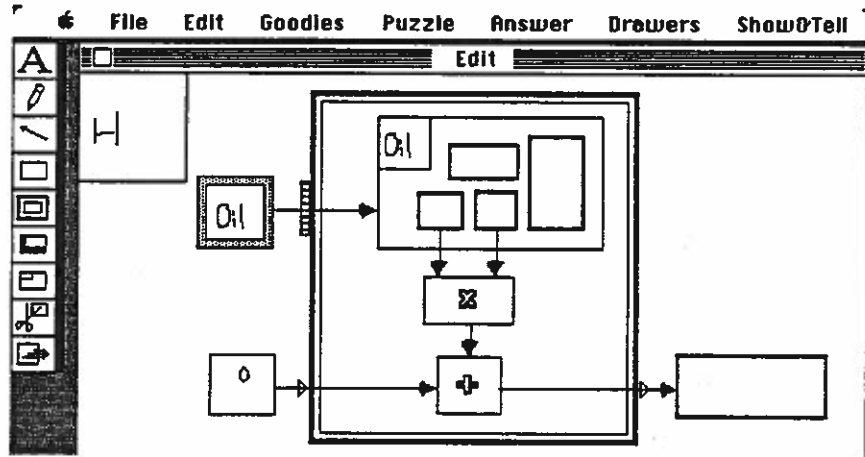


Figure 4.4

This puzzle goes through a database named "Oil" (figure 4.4a) and finds the total value of all of the stocks in the database. The file is feed through a parallel port into a structure box. Because the "Oil" icon was put in the name area of the structure box, all of the base boxes for the "Oil" database appear in it. The value of a particular stock is its price times the number of shares that are owned. This figure is calculated by explicitly multiplying the two fields in the database. Since this is done for each stock in the database, when the puzzle is completed, the total portfolio value will be calculated.

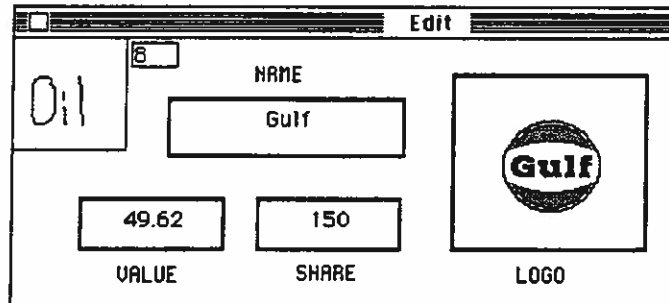


Figure 4.4a

### § 4.3 Editing Icons

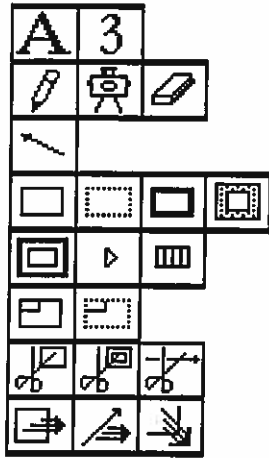


Figure 4.5

#### General Remarks

This group of icons (figure 4.5) is used for creating and editing Show and Tell puzzles. There is not enough room in the editing bar to show all of the editing tools at once so the tools have been grouped together to save space. To select a tool that is not currently shown, move the pointer to the portion of the black bar that is next to an icon in the same group as the one you wish to use and then press the mouse button. A sidebar appears showing all the icons in that group. Drag the pointer to the icon you wish to select and release the button.

The explanations of the icons below are arranged by group. All of the icons in a particular group are listed at the beginning of the section then each icon in the group is explained.

**Note:** Changes made to a puzzle during an editing session are not automatically saved. To save the changes, drag the icon from the current puzzle area into the drawer in which you wish to save the puzzle.



These icons are used to enter text or numbers into the puzzle. Show and Tell uses the Geneva 12 font.



#### Text Icon

Clicking on this icon lets you enter text into a name area, the background, or a box. If you type text that represents a number with the text tool, Show and Tell will automatically convert it to a number. The backspace key may be used to erase a character. A click of the mouse will end the text entry.



#### Numeric Icon

Clicking on this icon allows you to enter a number into the name area, the background, or a box.



This group of icons represent tools that are used to create and change images in the background and the name area. These tools do not affect the box structure of the puzzle.



#### Pencil

Changes the pointer to the pencil allowing you to draw in the name area or the background.



## Camera

Clicking on this icon opens the clipboard and allows you to paste from the clipboard into the name area or into the background.

To paste into the name area:

- \* Select the camera.
- \* Click in the name area.

The clipboard window will appear.

- \* Select the closed box from the editing bar.
- \* Draw a box around the portion of the clipboard that you want in the name area.
- \* Close the clipboard window.

To paste into the background:

- \* Select the camera.
- \* Click in the background. The point that you click in the background will be the top left corner of the image to be pasted. The clipboard will appear.
- \* Select the closed box from the editing bar.
- \* Draw a box around the portion of the clipboard that you want in the background.
- \* Close the clipboard window.



## Eraser

Clicking on the icon will change the pointer to the eraser allowing the user to erase in the name area, the background, or a puzzle.

Double clicking on the icon will clear the background of the puzzle window.



## Arrow

Selecting this icon lets you draw arrows. Only arrows that connect two boxes are considered part of the box structure.

To draw an arrow between:

- \* Select the arrow icon.
- \* Click where the tail of the arrow should be.
- \* Move the pointer to the desired terminating point and double click.



These tools allow you to draw boxes that will be part of the box structure of a puzzle.

To draw a box:

- \* Select the the icon depicting the type of box you wish to draw
- \* Move the pointer to a location where one of the corners of the box will be
- \* Drag the pointer to the diagonaly opposite corner of the box and release the button

You may continue to draw boxes of the selected type without having to reselect the icon again.



These tools allow you construct iteration boxes and to connect the appropriate serial and parallel ports to them.



#### Iteration box tool

This tool lets you draw the iteration box on the screen. Drawing an iteration box is the same process as drawing any other box in Show and Tell.



#### Serial port tool

Use this tool to attach serial ports to an iteration box. To put a pair of ports on the iteration box, select the place along the edge of the iteration box you wish to place an incoming serial port and click the mouse.

To use a serial port, draw an arrow going through the triangle.



#### Parallel port tool

Use this tool to add a parallel port to an iteration box. Parallel ports are made like boxes. To use the port, draw an arrow going through the port.



The scissors allow the user to cut various parts of the the box structure. They may not be used to erase any part of the background or name area.



#### Box scissors

These scissors are used to delete a single box from the box structure of the puzzle.

- \* Select the **Box-scissors**.
- \* Place the pointer on the box you wish to delete.
- \* Click the mouse button.

You may continue to remove boxes from the puzzle without reselecting the icon.



### Multiple box scissors

This tool is used to delete a box and all boxes contained it contains. Arrows that are in the box are not deleted.



### Arrow scissors

This tool is used to delete arrows from the box structure of a puzzle. To delete an arrow:

- \* Select the tool by clicking on its icon.
- \* Place the pointer on the arrow you wish to delete.
- \* Click the mouse button to delete the arrow.



These tools are used to reposition and change the size of boxes and arrows.



### Box moving tool

This tool allows you to move boxes.

- \* Select the icon.
- \* Place the pointer in the box you wish to move.
- \* Drag the box to its new location.

Show and Tell will not let you drag a box so that it overlaps a portion of another box. You may, however, move the box so as to completely contain another box.



### Arrow Moving Tool

This tool allows you to move arrows.

- \* Select the icon
- \* Place the pointer on or near the arrow you wish to move
- \* Drag the arrow to the new location



### Box Expansion Tool

This tool allows you to enlarge or decrease the size of a box in the box structure.

- \* select the icon
- \* place the pointer on the lower right corner of the box
- \* drag the corner until the box is the desired size

You will not be allowed to change the size in such a way as to partially overlap another box. The box expansion tool may also be used to change the size of the name area.

## § 4.4 Show and Tell Menus

File	
New	
Open	
Save	
Save Into...	
Display	
Print	
Print Screen	
.....	
Switch	
.....	
Quit	⌘Q

### FILE

The **FILE** menu has selections that let you open and close files and puzzles. This is also where you exit from Show and Tell. The items in the menu perform operations only on specific objects. Listed under many items of the menu are several objects. The command operates on each of the objects in the way specified.

#### New

Opens a new window for editing.

#### Save (Not implemented)

Save the selected file in the selected drawer. If no drawer is specified, then the parent drawer is assumed.

#### Save Into....(Not implemented)

This command is just like the **SAVE** command except that a dialogue box is opened allowing you to save the selected item with a different name.

#### Display (Not Implemented)

##### Drawer

Display all of the files in the selected drawer.

##### File

Display all of the known solutions of a file.

#### Print (Not Implemented)

Print the selected puzzle.

#### Print Screen (Not Implemented)

Prints the the screen onto the printer.

#### Switch (Not Implemented)

Ejects the disk in the external drive.

#### Quit

Quits the Show and Tell language and brings you back to the finder.



**Edit**

<b>Undo</b>	
<b>Cut</b>	<b>⌘H</b>
<b>Copy</b>	<b>⌘C</b>
<b>Paste</b>	<b>⌘Z</b>
<b>Clear</b>	

**Goodies**

<b>Grid</b>	
<b>Hairline Cursor</b>	
<b>Clean Up</b>	
<b>Order Base Boxes</b>	
<b>Memory Status</b>	
<b>Help</b>	
<hr/>	
<b>Precision 0</b>	
<b>Precision 2</b>	
<b>✓Precision 6</b>	
<b>Precision 10</b>	

**Edit**

**Undo** Not implemented

**Cut** (Used only with the Camera)  
Cuts the selected items from a desk accessory (e.g. the scrapbook) and puts them on the clipboard.

**Copy** (Used only with the Camera)  
Copies the selected items from a desk accessory and puts them on the clipboard.

**Paste**  
Only used by desk accessories.

**Clear**  
Erases the box-structure from the selected window.

**Goodies Menu**

**Grid**  
The grid is an invisible grid that is superimposed on the editing window and which allows you to draw arrows that line up. If the grid is off, selecting this item will turn the grid on. If the grid is on, selecting this item will turn the grid off.

**Hairline Cursor**  
The hairline cursor changes the pointer to a pair of intersecting lines that extend across the screen. These lines may be used to align boxes and arrows while editing. To turn the hairline cursor off, select **Hairline cursor** again.

**Clean Up**  
Redraws the selected box-structure eliminating arrows that do not connect two boxes. All other arrows are trimmed to fit directly against the sides of the box.

**Order Base Boxes**  
Displays the ordering index of the base boxes of a puzzle. This is useful in figuring out the association between the formal and actual parameters. **Reset** clears the index numbers.

**Memory Status**  
Shows the memory allocation of the system. (For system debugging)

**Help** (Not Implemented)

**Precision**  
The precision commands are used to change the precision of arithmetic operations. Precision 2 means that calculations are carried out with two decimal places. The current precision is indicated by a check mark next to the degree of precision.

**Puzzle**

Solve  
Pause  
Continue  
Halt

Reset

Set Stop Mark  
Clear Stop Mark

**Answer**

Enter  
Find  
Find Next

Buttons  
Clear Base Boxes

**Puzzle Menu**

This menu contains the commands to tell STL when to start and stop looking for solutions to a selected puzzle.

**Solve**

Show and Tell will start solving the currently active puzzle. Show and Tell is able to solve several puzzles at the same time.

**Pause**

Stops Show and Tell from completing the puzzle in the currently active window. Completion may resume with the **Continue** function.

**Continue**

Causes Show and Tell to resume work on a puzzle that has been paused. The puzzle must be in the currently active window.

**Halt**

Show and Tell stops execution of the puzzle. There is no way to resume execution after halting.

**Reset**

All of the boxes that contain system completed information are blanked.

**Set Stop Mark**

Put a stop mark at the selected box. A box may be selected by holding down the Shift key and clicking in the box. During completion, the system will pause before executing a box with a stop mark (break point).

**Clear Stop Mark**

Clear previously set stop marks.

**Answer Menu**

This menu has the commands for searching database puzzles. All of the commands work with the currently displayed puzzle.

**Enter** (Not Implemented)

**Find**

Find solution in database meeting the constraints of the puzzle.

**Find Next**

Find the next solution in the database meeting the constraints.

**Buttons**

Selecting this item will display the buttons on the screen. The buttons are used to enter solutions into a puzzle or to search for solutions already in a puzzle.

**Clear Base Boxes**

Clears the base boxes of the current puzzle. This allows you to enter a new solution into a database, for example.

**Drawers**

**Input Output**  
**Arithmetic**  
**Miscellaneous**

**New**  
**Open**  
**Throw Away**

**Show&Tell**

**Place Call**  
**Answer Call**  
**Hang Up**

**DRAWERS MENU**

This menu lets you open a drawers, create a new drawer, or delete a drawer you no longer want. For more information about each of the icons in the three system defined drawers, see § 4.5 on system drawers.

**Input/Output**

Opens the Input/Output drawer.

**Arithmetic**

Opens the Arithmetic drawer.

**New**

Opens a new Drawer. A dialogue box will open asking for the name of the new drawer.

**Open**

Displays a dialogue box that shows the drawers on the current disk and allows you to change disks.

**Throw Away**

Delete a drawer from the disk. A dialouge box is opened allowing the user to specify which drawer to delete.

**SHOW & TELL**

This menu lets you communicate in picture with your friend on another Macintosh connected to your Macintosh by the AppleTalk local area network.

**Place call**

Initiate a picture phone call.

**Answer call**

Answer a picture phone call when your Mac beeps. It will open a phone window on your screen and the caller's screen. You can type any text or can draw by pencil. To change the position of text entry, click the mouse with the command key at the new cursor position. To change pencil to erasor, click the mouse with the option key. You can switch back to pencil by clicking the mouse again with the option key.

**Hang Up**

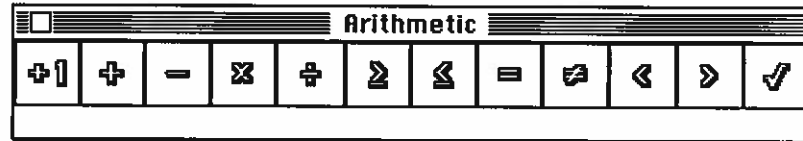
To terminate picture phone conversation. Either caller or callee can terminate conversation.

## § 4.5 System Drawers

Show and Tell provides three drawers of puzzles that have system defined functions. These puzzles may be used as subroutines in any puzzle.

### § 4.5.1 Arithmetic Drawer

The arithmetic drawer contains icons that provide the computational and logical powers of Show and Tell. There are two kinds of arithmetic icons; functional and relational icons. The functional icons take arguments as input and produce an output result. The relational icons take arguments as input and depending upon the relation between the inputs, become either consistent or inconsistent.



The Arithmetic Drawer

Unless otherwise noted, all of the functional icons take two inputs and produce one output value.



#### Increment Function

This icon represents a function of one input which returns the input value plus one.  $(x+1)$



#### Addition Function

The value of the function is the sum of the two inputs.  $(x+y)$



#### Subtraction Function

The value of the function is the second input subtracted from the first input.  $(x-y)$



#### Multiplication Function

The value of the function is the product of the two input values.  $(x*y)$



#### Division Function

This icon divides the first input by the second input.  $(x÷y)$



#### Greater than or equal Relation

This relation is consistent if the value of the first input is greater than or equal to the value of the second input, and inconsistent otherwise.  $(x≥y)$



#### Less than or equal Relation

This relation is consistent if the value of the first input is less than or equal to the value of the second input, and inconsistent otherwise.  $(x≤y)$



**Equality Relation**

This relation is consistent if the two input values are equal, and inconsistent otherwise. ( $x=y$ )



**Inequality Relation**

This relation is consistent if the two input values are unequal, and inconsistent otherwise. ( $x\neq y$ )



**Less than Relation**

This relation is consistent if the first input is less than the second input, and inconsistent otherwise. ( $x < y$ )



**Greater than Relation**

This relation is consistent if the value of the first input is greater than the value of the second input, and false otherwise. ( $x > y$ )



**Square Root Function**

This function takes only one input and returns the square root of the input value. If the input is negative, the function becomes inconsistent. ( $\text{sqr}(x)$ )



**Decrement Function**

This function takes only one input and returns the input minus one. ( $x-1$ )

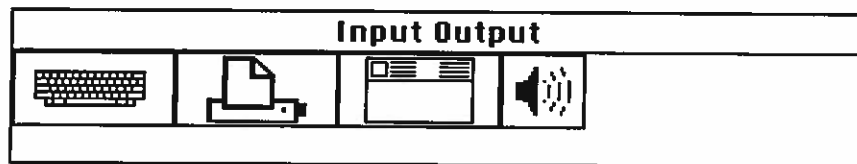


**Greater than zero Relation**

This relation takes only one input value and is consistent if the input value is greater than zero, and inconsistent otherwise. ( $x > 0$ )

**§ 4.5.2 Input/Output Drawer**

This drawer contains icons representing various means for input and output in Show and Tell. An arrow pointing to an output icon will send information to the indicated device. Likewise, an arrow from an input icon will enter information from the designated source.





### Keyboard

This operation requires one text input. It opens an input window with the text as the name. Any text entered into the input window until the window is closed is the output of this operation. If the text start with a number, the entire text is considered as a number.



### Printer Icon

Any text or number can be printed but not an image.



### Window

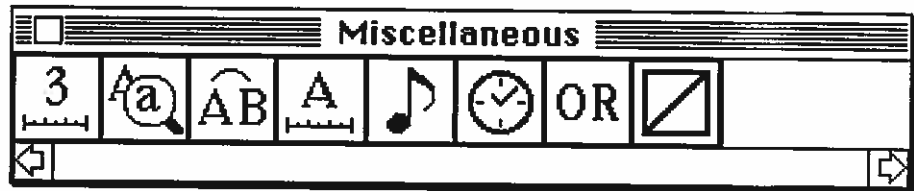
During completion of a puzzle, this icon opens a window named **Solution** which displays all values sent to it. The name of the window can be specified by adding it as the first argument to the box. If several window icons are given the same name, all values sent to those window icons will be displayed in a single window of the given name.



### Speaker

The contents of any box that points to the speaker icon are spoken through the Macintosh internal speaker.

## § 4.5.3 Miscellaneous Drawer



This group of icons represent puzzles that manipulate and examine text, do the logical operations of Not and Or, and do system calls such as sounding the chime and displaying the time.

**Numeric Precision**

This icon sets the number of decimal points that are displayed in the puzzle. It takes a single input which is the number of decimal places.

**Text Search**

This icon looks for a string in a text. If the string is found, then the puzzle is consistent. If the string is not in the text, the puzzle is inconsistent. The search does not distinguish between upper and lower case. The first input is the text to be searched, the second input is the string.

**Concatenation**

This puzzle concatenates two strings of text. The second input string is appended to the first. This puzzle only works on strings of text.

**Text Length**

This icon takes text as its input and produces the number of characters in the text as output. Blanks etc. are counted as well as alphanumeric characters.

**Chime**

This icon sounds the Macintosh chime. It takes a single numeric value as the input. The value of the input determines the duration of the chime.

**Clock**

This icon produces the current time as its output. The output is a text string of the form: 'HH:MM:SS AM' (or PM) (e.g. 10:15:03 PM)

**Logical Or**

This icon is consistent if at least one of its inputs has a value. This icon can accept any number of inputs.

**Logical Not**

This puzzle is consistent as long as none of its inputs has a value. As soon as it has an input, it becomes inconsistent.