Macaroni

Presenting Macaroni

Abstract

Macaroni is an interactive training tool for helping people learn the Macintosh Toolbox.

It gives curriculum developers a medium for writing interactive courseware about Macintosh programming.

This manual presents Macaroni and shows how it works.

 Apple Computer, Inc.

 Apple Computer, Inc.

Adam Bartlett, Rich Millet and Jonathan Zar wrote Macaroni with support, technical assistance and/or thoughtful criticism from the following people:

Harvey Alcabes
Bill Atkinson
David Barroso
Mary Boetcher
Gary Bond
Debra Brackeen
Chris Brown
Dallas Brown
Tom Chavez
Laurel Frishman
Moira Gagen
Mitchell Gass
Jean-Louis Gassee
Dennis Gibbs
Deborah Grits
Kathy Haranzo
Cimon Hartley
Eric Hulteen
David Hwu
Steven Johnson
Kristee Kreitman
Sioux Lacy
David Leffler
Jordan Mattson
John Mitchell
Anne Nicol
Jacqueline Perez
Maureen Roddy
Lew Rollins
John Schmidli
Alan Sun
Wendy Tajima
Eileen Unruh
Nick Vaccaro
Dan Winkler
Joe Zuffoletto

Revision 5/89 ß2.2 Draft

Table of Contents

Overview

This manual is an introduction to Macaroni—the interactive tool for learning the Macintosh Toolbox.

The first part of this manual, "Overview", answers the following basic questions:

• What is Macaroni?
• What do I need to run it?
• What's new in the latest release?

The second part of this manual, "Ingredients", presents the Macaroni user interface.

The third part of this manual, "Techniques", shows how to use the interface. It demonstrates how to execute calls and how to save calls into a script.

The fourth and final part of this manual, "Basic Recipes", guides you through a simple tutorial, *PointerPlay*, that demonstrates Macaroni in action.

The authors anticipate that students will be supplied with specific training materials when they are using Macaroni as part of a live lecture series, training course, self-paced training module, audio training program or video.

The information in this manual should be used in conjunction with and not in lieu of any specific training materials that are supplied by an instructor.

Macaroni Is A Training Tool

Macaroni is an interactive training tool for helping people learn the Macintosh Toolbox.

It gives curriculum developers a medium for writing interactive courseware about Macintosh programming.

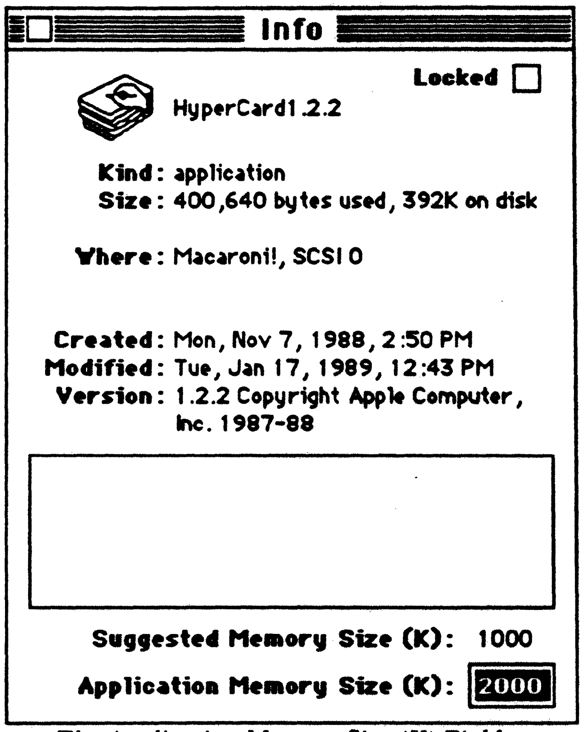
Macaroni visualizes, in a simple and direct manner, the most popular calls from Apple's classic reference Inside Macintosh.

Calls are presented as fill-in-the-blank forms in an exploratory environment that's fun to use. The forms can be scripted and visualized during execution.

Prerequisites Before Using Macaroni

Macaroni can run on any Macintosh with a hard disk drive that is able to run HyperCard. The hard disk must have at least one megabyte of free space and it must have HyperCard (version 1.2.2 or later) and a Home stack installed anywhere on the disk within a common folder.

Macaroni performs best in more than one megabyte of RAM, at least *two megabytes is strongly recommended.* If you use MultiFinder and you have extra RAM, edit the "Application Memory Size (K)" field for the HyperCard application to take full advantage of the RAM before you run Macaroni. The "Application Memory Size (K)"field is found in the HyperCard Get Info window as shown in the following picture:



HyperCard1.2.2

Kind: application
Size: 400,640 bytes used, 392K on disk

Where: Macaroni!, SCSI 0

Created: Mon, Nov 7, 1988, 2:50 PM
Modified: Tue, Jan 17, 1989, 12:43 PM
Version: 1.2.2 Copyright Apple Computer, Inc. 1987-88

Suggested Memory Size (K): 1000
Application Memory Size (K): 2000

The Application Memory Size (K) Field

Macaroni runs faster on machines using a 68020 or 68030 processor, such as the Macintosh II, IIx, IIcx and SE/30.

The scripting language for Macaroni is HyperTalk, the easy to write language of HyperCard.

For information about HyperTalk, consult the online help system in HyperCard, any of the books on HyperTalk in your local bookstore, or the following Apple manual published by Addison-Wesley Publishing Company, Inc: *HyperCard Script Language Guide, The HyperTalk Language*. For information on how to order this manual from APDA, the Apple Programmers and Developers Association, call: (800) 282-2732.

Notes For Experienced Users

The Beta version of Macaroni released in conjunction with this manual is very different from all prior releases of Macaroni.

The most obvious change is an improved interface that is easier to use. The interface reflects a simplification in design.

The visual scripting methods for specifying flows of control and data that early users found difficult are gone. Gone too are the notions of "loading" and "point-to-point linking."

Numerous controls have been consolidated for a cleaner look and feel.

Scripting has been enhanced and scripts may now be created directly in easy to write HyperTalk.

RAM and hard disk storage are consumed more efficiently so that Macaroni can run on a one-megabyte SE.

Persistent drawing to a separate window is now supported. You can now watch an animated sequence of Quickdraw calls in the card window while viewing their effect in the drawing window.

Several new features requested by instructors have been added, including the ability to pause a script and resume its execution under mouse control.

Numerous bugs have been fixed.

All of these improvements have been made as the direct result of your feedback.

Please continue to provide criticism and reports of any problems that you discover.

Your bug reports and comments are greatly appreciated. Please send your AppleLinks to "Macaroni" and your letters to the following address:

Apple Computer, Inc.
20525 Mariani Avenue
Cupertino, CA 95014
Attn: Macaroni MS 27BC

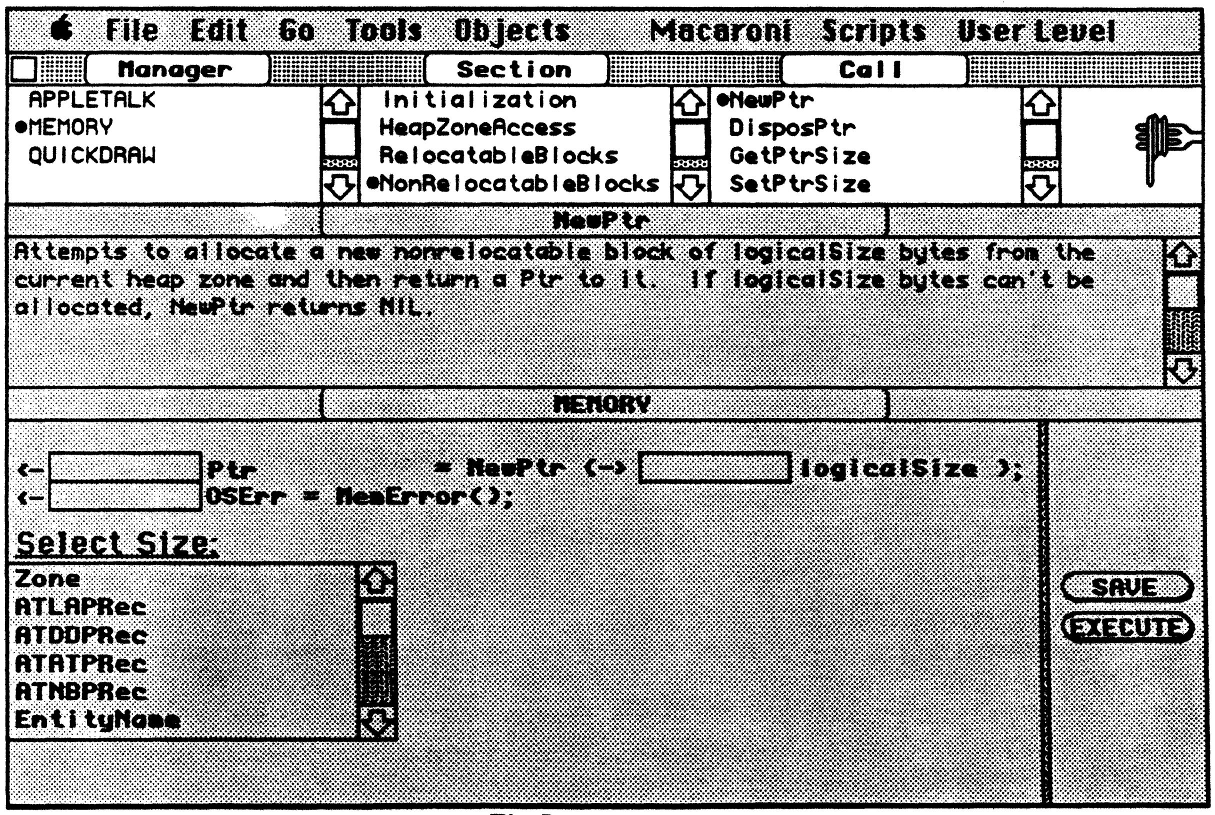
Ingredients

All of the major displays in Macaroni share a common interface. This makes it easy to find things. Macaroni provides a browser for finding calls and menus for finding everything else.

The next several sections of this manual introduce the ingredients that comprise the Macaroni interface.

The Browser

Below the menu bar in every display is a three-pane browser. Here is a picture showing the Browser highlighted for emphasis:



Attempts to allocate a new nonrelocatable block of logicalSize bytes from the current heap zone and then return a Ptr to it. If logicalSize bytes can't be allocated, NewPtr returns NIL.

```
<- [        ]Ptr      = NewPtr (-> [        ] logicalSize );
<- [        ]OSErr = MemError();
```

The Browser

The Browser is used to select a call or a datatype within a manager.

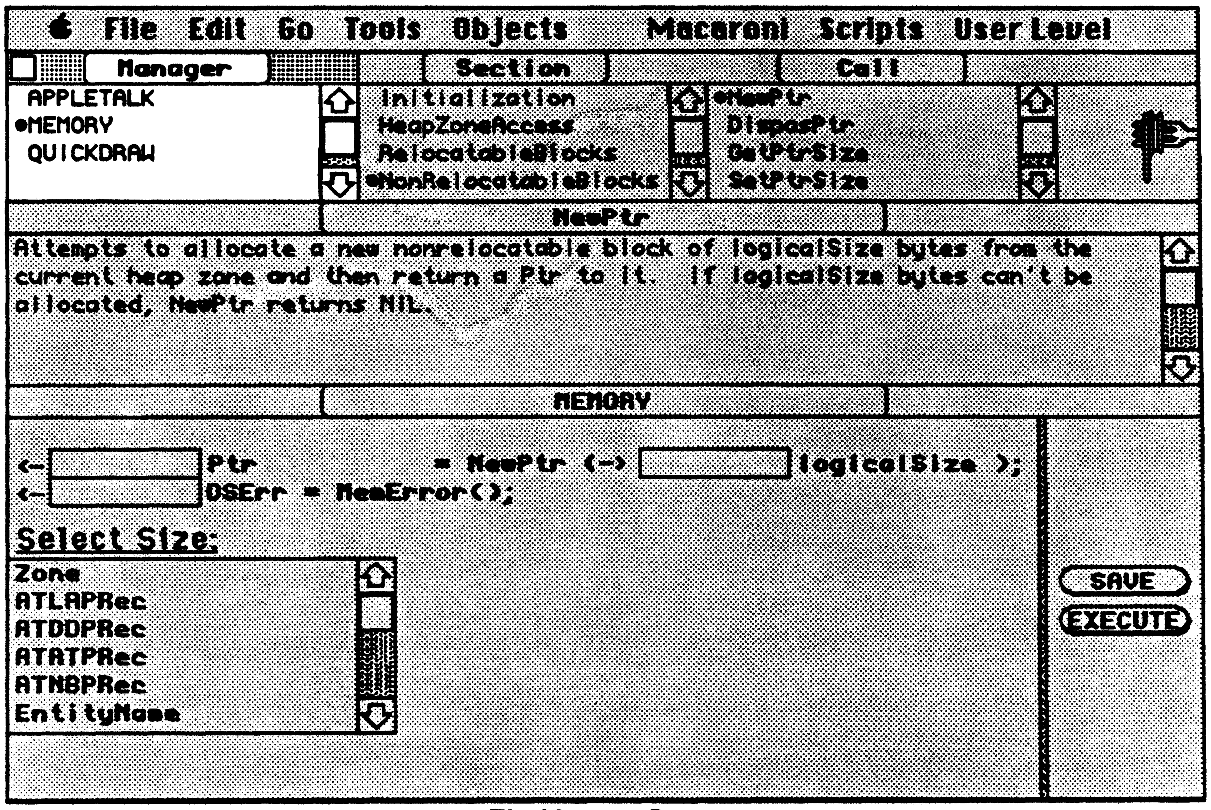
The picture above shows the selection of the *NewPtr* call in the *NonRelocatableBlocks* section of the *Memory* manager.

The Browser consists of a Manager Pane, a Section Pane, a Call Pane, and the Macaroni button.

On the far left of the browser is a close box for quitting Macaroni.

The Manager Pane

Here is a picture of the Manager Pane, highlighted for emphasis:



Attempts to allocate a new nonrelocatable block of logicalSize bytes from the current heap zone and then return a Ptr to it. If logicalSize bytes can't be allocated, NewPtr returns NIL.

```
<-[          ]Ptr        = NewPtr (->[          ]logicalSize );
<-[          ]OSErr = MemError();
```

The Manager Pane

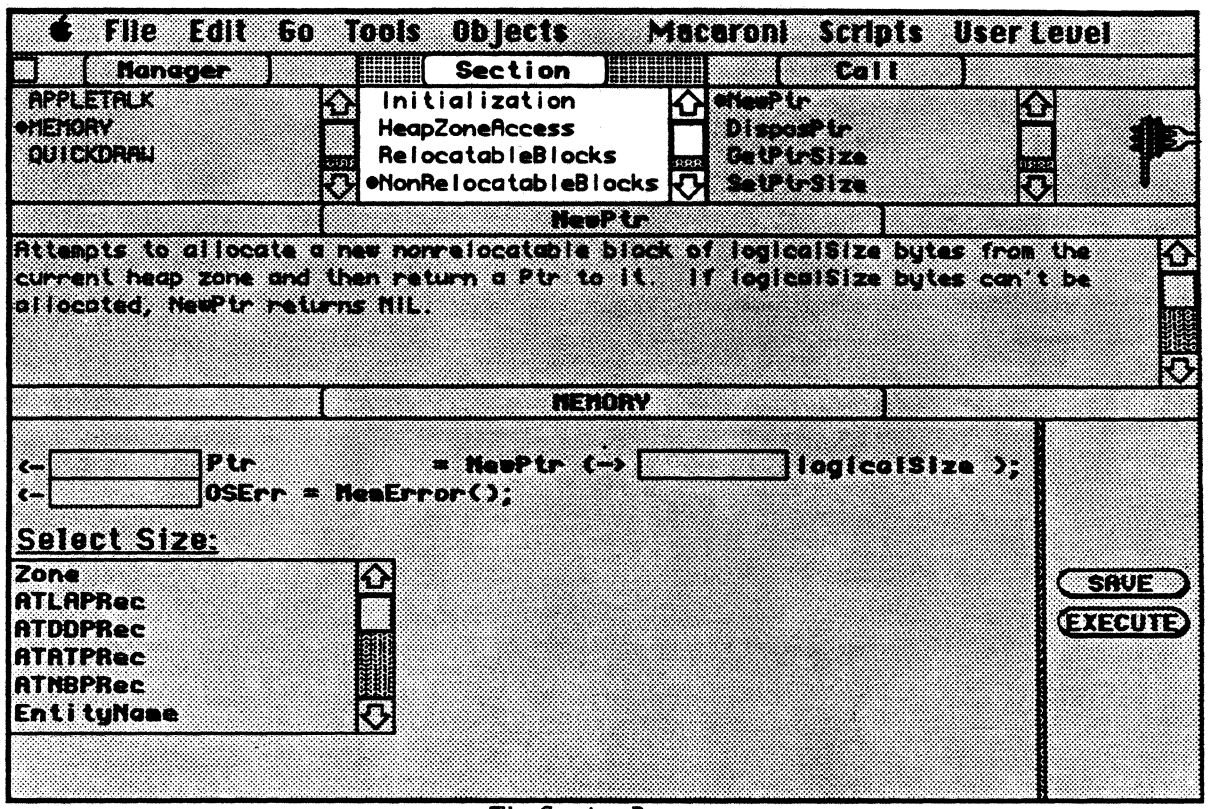
The names in the Manager Pane are managers in the Toolbox as documented in *Inside Macintosh.*

In the picture above the Manager Pane has been set to the *Memory* manager. (The Memory Manager allocates primary storage in RAM.)

The Section Pane

The Section Pane sits to the right of the Manager Pane.

Here is a picture of the Section Pane highlighted for emphasis:



Attempts to allocate a new nonrelocatable block of logicalSize bytes from the current heap zone and then return a Ptr to it. If logicalSize bytes can't be allocated, NewPtr returns NIL.

```
<-[          ]Ptr        = NewPtr (-> [          ]logicalSize );
<-[          ]OSErr = MemError();
```

The Section Pane

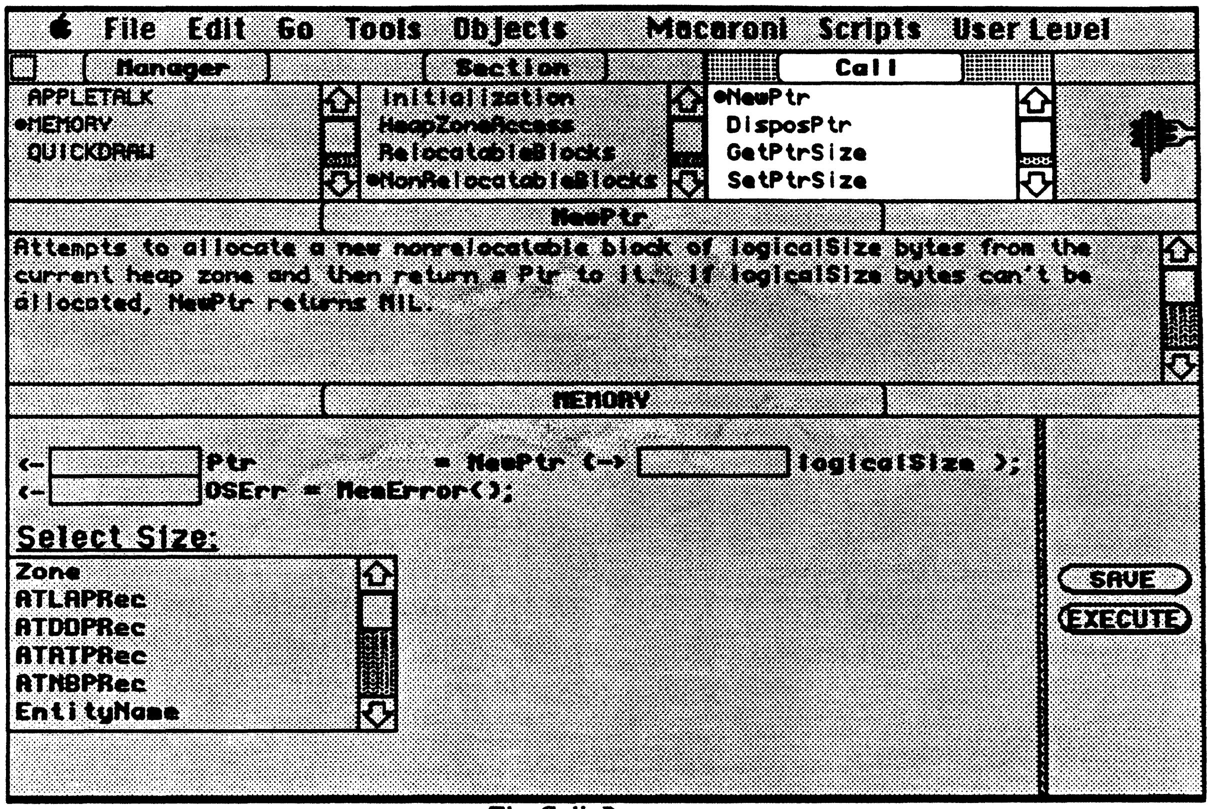
The Section Pane holds a list of the names of each of the sections in the manager that is selected in the Manager Pane. A section is a part of a manager that organizes several related calls.

In the picture above the Section Pane has been set to the *NonRelocatableBlocks* Section of the *Memory* manager. (This section of the Memory Manager controls the allocation of nonrelocatable storage.)

The Call Pane

The Call Pane sits to the right of the Section Pane.

Here is a picture showing the Call Pane highlighted for emphasis:



Attempts to allocate a new nonrelocatable block of logicalSize bytes from the current heap zone and then return a Ptr to it. If logicalSize bytes can't be allocated, NewPtr returns NIL.

```
<-[          ]Ptr      = NewPtr (-> [          ] logicalSize );
<-[          ]OSErr = MemError();
```

The Call Pane

The Call Pane holds a list of calls from *Inside Macintosh* that are found in the section of the manager identified in the Section and Manager panes. A call within a manager is a trap to a piece of code provided by Apple.

In the picture above the Call Pane has been set to the *NewPtr* call in the *NonRelocatableBlocks* Section of the *Memory* manager. (This call allocates a block of storage addressed by a pointer.)

The Macaroni Button

To the right of the browser is the Macaroni Button:

The Macaroni Button causes the current script to be displayed.

When a script is selected the name of a script appears above the Macaroni button.

The Toolbox Pane

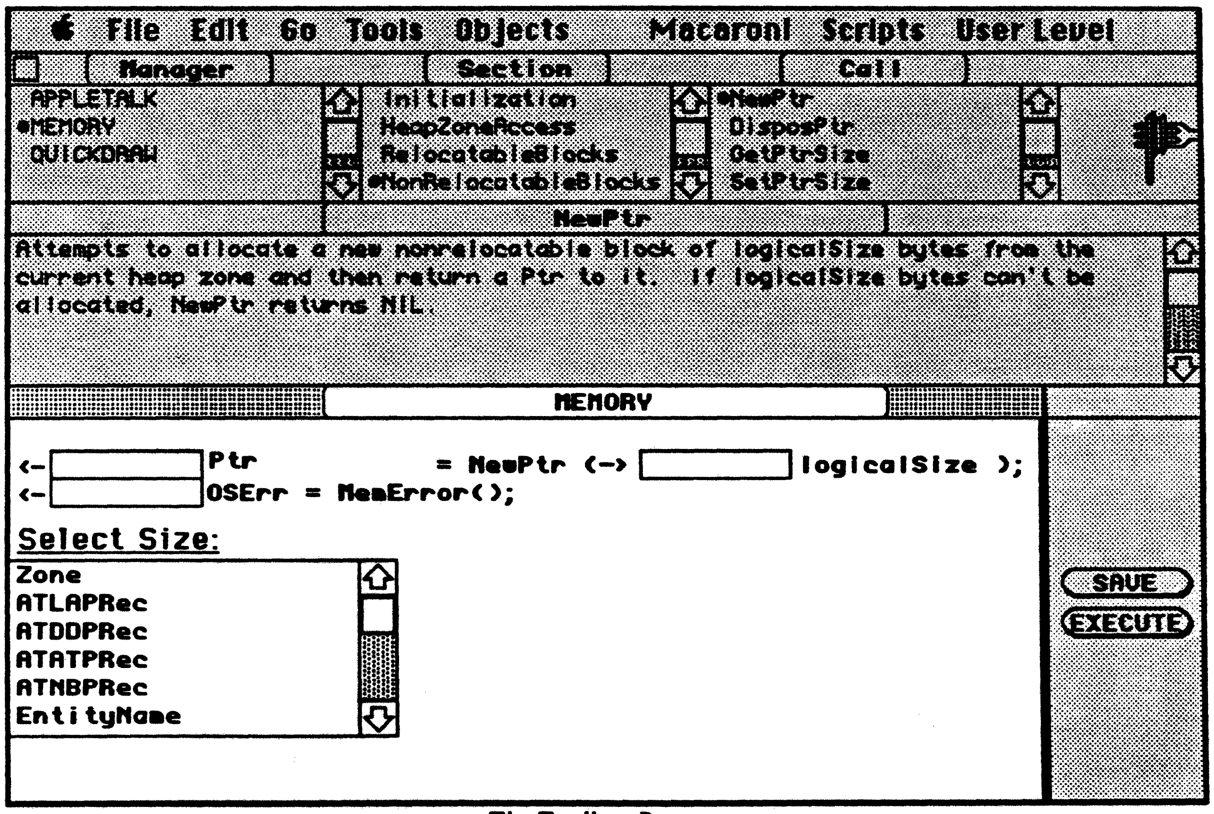
The Toolbox Pane presents a call, the parameters for the call, and the results of the call as an interactive fill-in-the-blank form.

In Macaroni, datatypes are accessed via calls that set or get the datatype.

The following two sections of this manual, "Toolbox Calls" and "Toolbox Datatypes", show how the Toolbox Pane is used.

Toolbox Calls

Here is a picture showing the Toolbox Pane highlighted for emphasis:



Attempts to allocate a new nonrelocatable block of logicalSize bytes from the current heap zone and then return a Ptr to it. If logicalSize bytes can't be allocated, NewPtr returns NIL.

```
<-[          ]Ptr        = NewPtr (->[          ] logicalSize );
<-[          ]OSErr = MemError();
```

The Toolbox Pane

The picture above shows the Memory Manager *NewPtr* call.

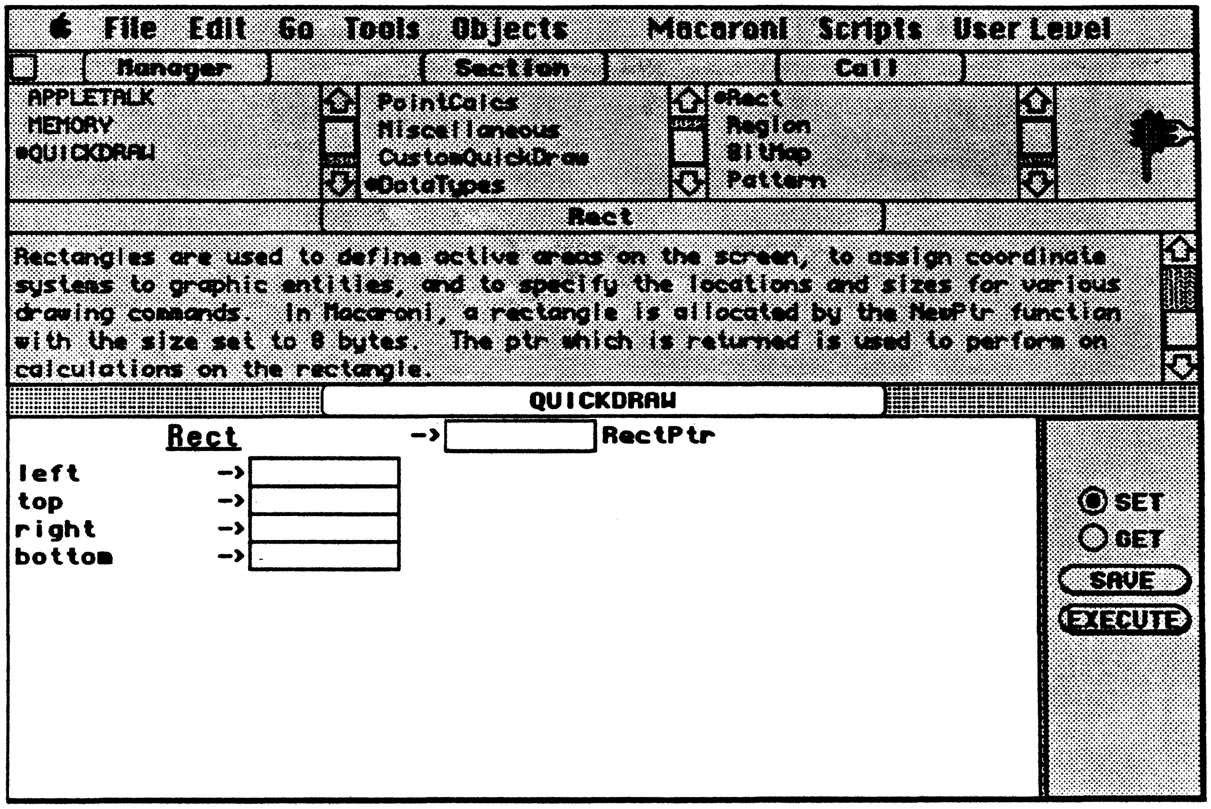
NewPtr takes one input parameter, *logicalSize*, and returns one output result, *Ptr*. The *NewPtr* call allocates a block of memory of length *logicalSize* and returns a pointer.

Macaroni always calls *MemError* after every other Memory Manager call. MemError has no input parameters. It returns one output result, *OSErr*. The *MemError* call checks for any errors and returns either a zero or an error code.

The list of logical sizes of popular data structures under the title <u>**SelectSize**</u> is a convenience provided by Macaroni for Memory Manager calls only. Clicking on the name of a data structure in the list puts the size of the data into *logicalSize*.

Toolbox DataTypes

Here is an example taken from the QuickDraw manager that shows the Rectangle datatype called "Rect":



Rectangles are used to define active areas on the screen, to assign coordinate systems to graphic entities, and to specify the locations and sizes for various drawing commands. In Macaroni, a rectangle is allocated by the NewPtr function with the size set to 8 bytes. The ptr which is returned is used to perform on calculations on the rectangle.

Rect -> RectPtr
left ->
top ->
right ->
bottom ->

[✓ SET] GET

A Datatype In The Toolbox Pane

The fill-in-the-blank box named **RectPtr** holds space for a pointer to a rectangle.

The boxes labeled **left,top,right** and **bottom** hold space for the values of the sides of the rectangle addressed by **RectPtr**.

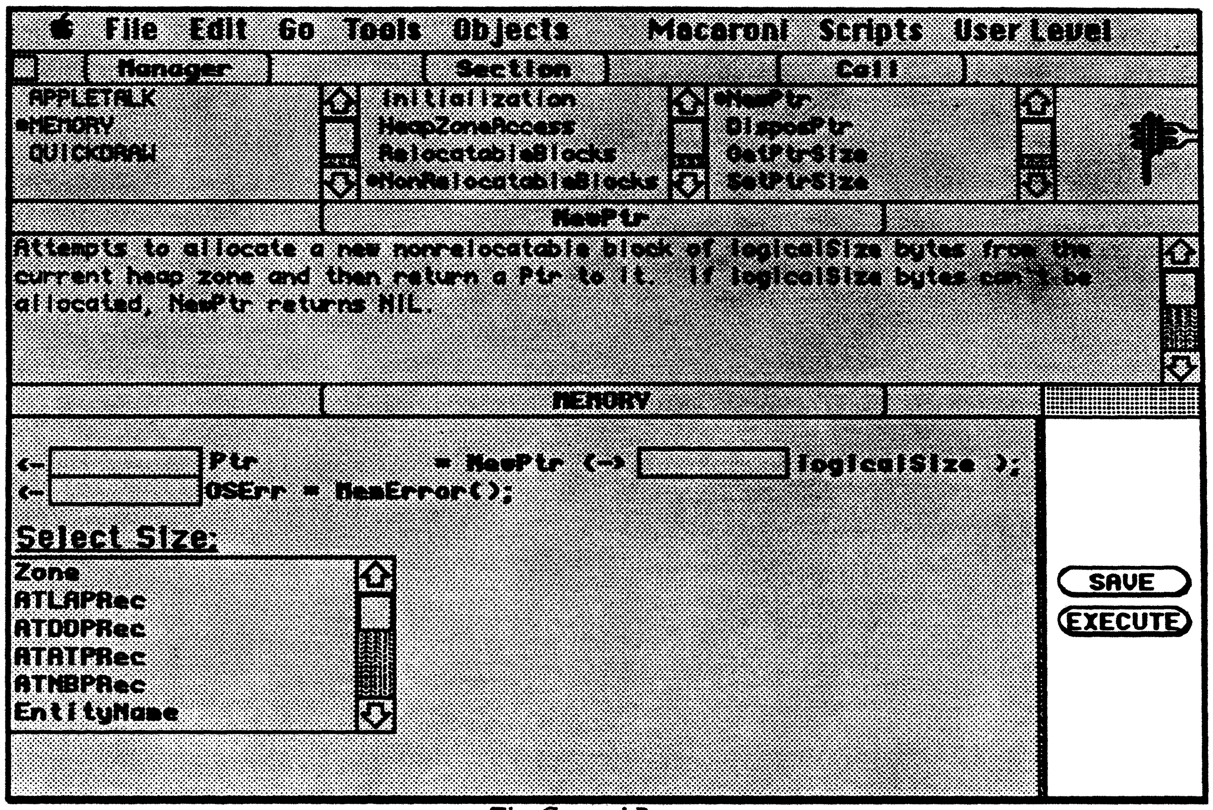
To set the values of a rectangle: put a pointer into the box labeled **RectPtr**, fill in the the boxes labeled **left,top,right** and **bottom**, select **SET** and press the **EXECUTE** button, in the Control Pane. (The Control Pane is shown on the following page of this manual.)

To get the values of a rectangle: put a pointer into the box labeled **RectPtr**, select **GET**, press the **EXECUTE** button and watch the top, left, bottom and right sides of the rectangle appear in the boxes labeled **left,top,right** and **bottom**.

The Control Pane

To the right of the Toolbox Pane is the Control Pane.

Here is a picture showing the Control Pane highlighted for emphasis:



Attempts to allocate a new nonrelocatable block of logicalSize bytes from the current heap zone and then return a Ptr to it. If logicalSize bytes can't be allocated, NewPtr returns NIL.

```
<-          Ptr          = NewPtr (->          logicalSize );
<-          OSErr = MemError();
```

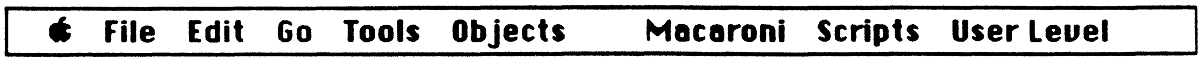
The Control Pane

The Save button (SAVE) saves the call (that is displayed in the Toolbox Pane) into the script that is checked in the Scripts menu.

The Execute button (EXECUTE) executes the call that is displayed in the Toolbox Pane.

The Menu Bar

The Menu Bar in Macaroni has three menus—Macaroni, Scripts, and User Level— that augment the standard HyperCard menus:

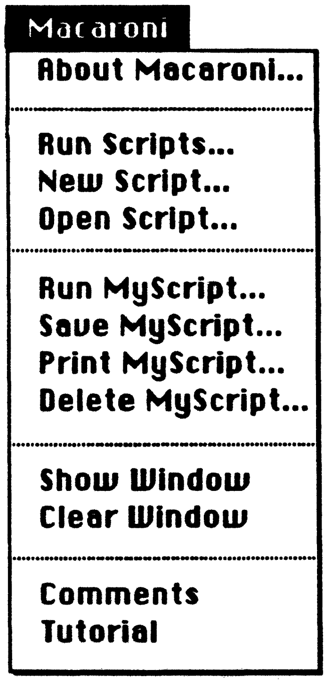


The Menu Bar

The next three sections of this manual, "The Macaroni Menu", "The Scripts Menu", and the "User Level Menu", explain the purpose and operation of each of the three Macaroni menus.

The Macaroni Menu

Here is a picture of the Macaroni menu as it would appear after a script called "MyScript" has been selected:



The Macaroni Menu

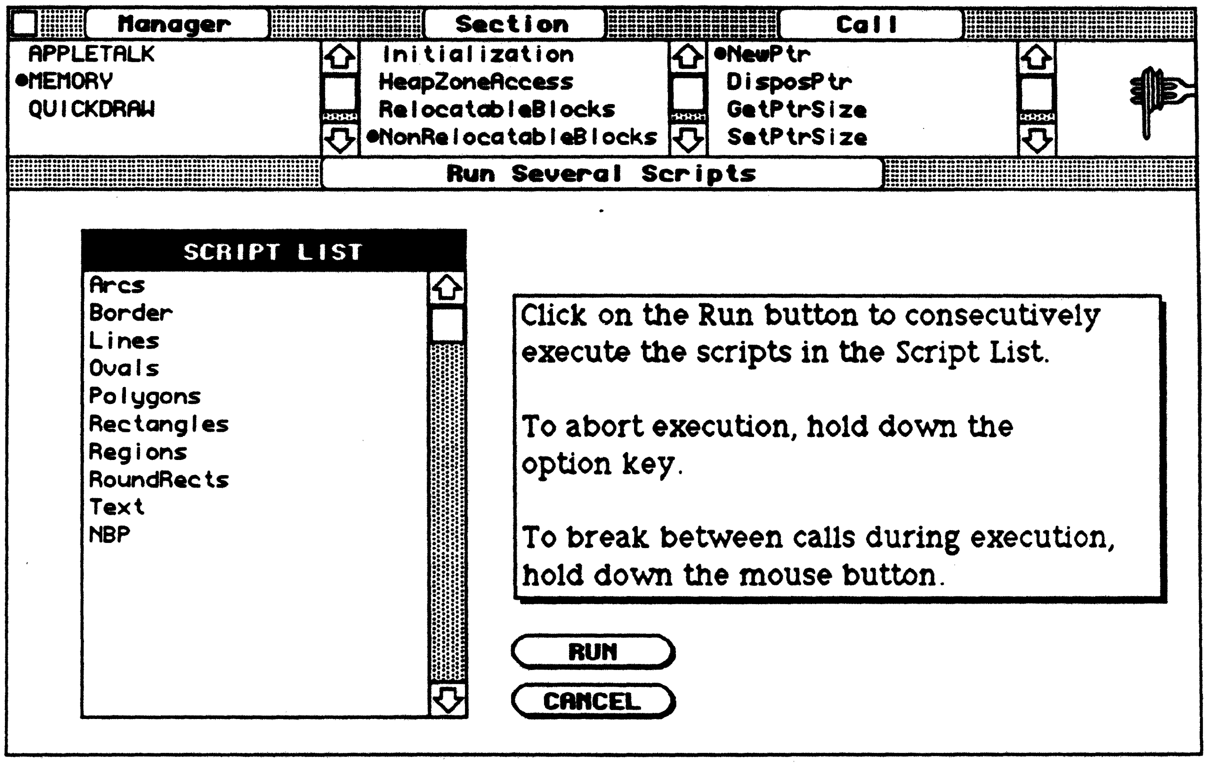
Here is an explanation of each of the items of the Macaroni menu:

Macaroni About Macaroni...

This command displays the Product Label.

Macaroni Run Scripts...

This command causes the Run Scripts dialog to be displayed.



Click on the Run button to consecutively execute the scripts in the Script List.

To abort execution, hold down the option key.

To break between calls during execution, hold down the mouse button.

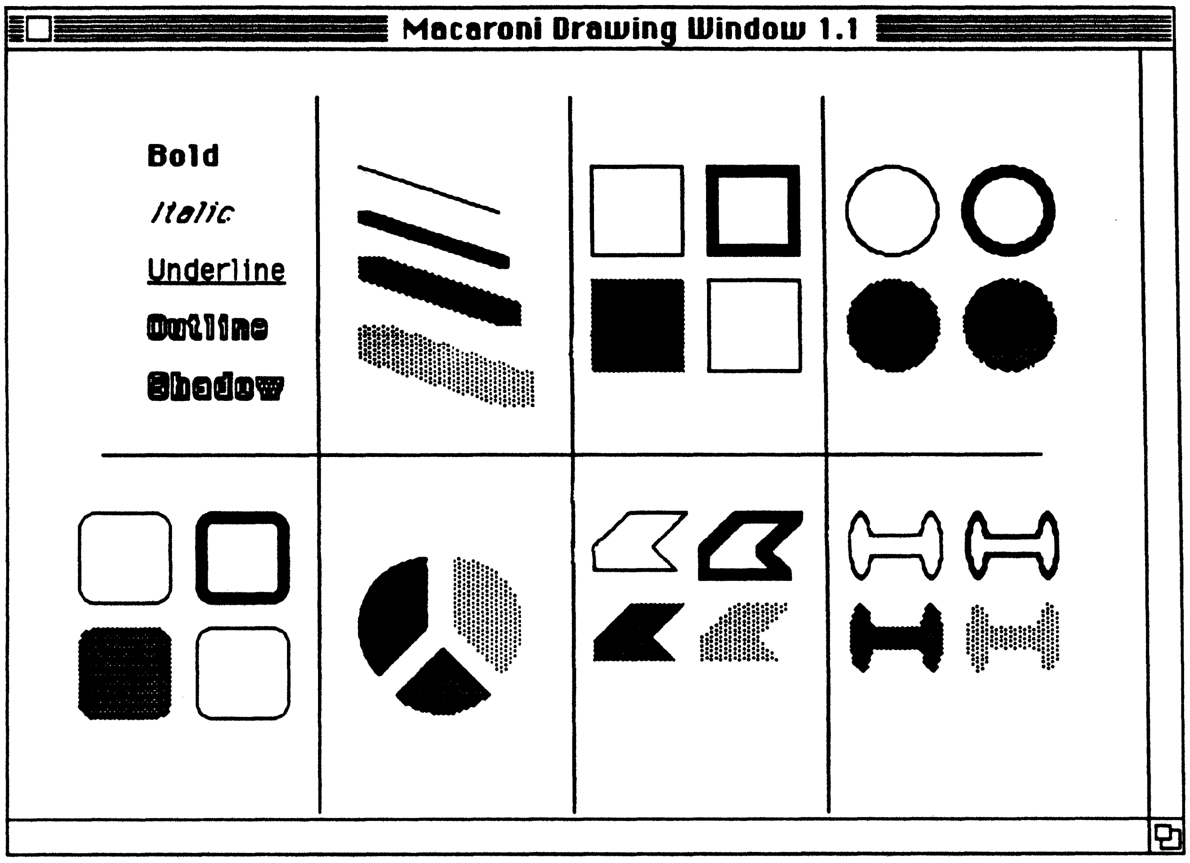
The Run Scripts Dialog

The Run Scripts dialog contains space for a list of scripts that are to be run one after another. It also contains a button to start running the scripts in the list and a button to cancel the dialog.

The list of scripts to be run one after another must be supplied by the user and must consist of one or more of the scripts named in the Scripts menu. Macaroni will halt if it encounters a name that is not in the Scripts menu.

If you click on the **RUN** button in Run Scripts dialog, and confirm your request, Macaroni will run each of the scripts and visualize each Toolbox call.

The following picture will be drawn:



The Drawing Demo

Macaroni New Script...

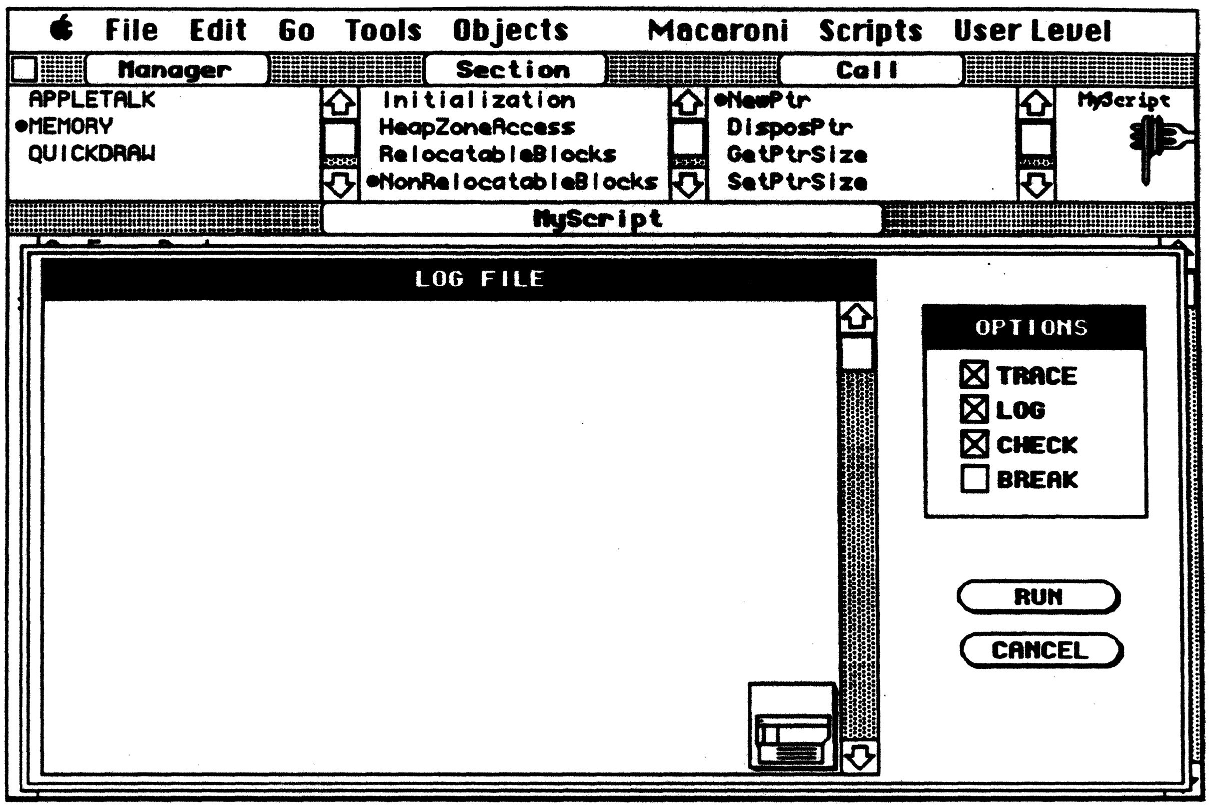
This command creates a new empty script. It puts up a dialog that asks for a name. The name entered into the dialog will become the name of the new script.

Macaroni Open Script...

This command opens a text file and recreates a Macaroni script from the file. It puts up a standard Macintosh SFGet dialog box for selecting the file to be opened.

Macaroni Run MyScript...

This command displays the Run dialog:



☒ TRACE
☒ LOG
☒ CHECK
☐ BREAK

The Run Dialog

To run a script from the dialog, select the options you want and press the Run button.

To exit the dialog and return to the current script, press the Cancel button.

To abort a running script, hold down the Option key until Macaroni safely halts a script at the start of a new call.

Don't abort HyperTalk using the Command key: ⌘

Here is a summary of the option settings:

☒ **TRACE**
The Trace option, when checked, causes Macaroni to trace the flow of control by visualizing each Toolbox call. Normally, this option should be checked. If you want to get the results of a script without visualizing each call, don't check the Trace option.

☒ **LOG**
The Log option, when checked, causes Macaroni to log the results of a script in the part of the Run dialog entitled "Log File". After Macaroni has run you can print the log by pressing the button that looks like a LaserWriter:

The Log File Print Button

☒ **CHECK**
The Check option, causes Macaroni to attempt to validate each call's parameters by type and range before a call is executed.

☐ **BREAK**
The Break option, when checked, causes Macaroni to pause after every call and ask if you want to continue with the next call. Normally, this option should *not* be checked.

You can always force Macaroni to pause between calls, even if the Break option is unchecked, just by holding down the mouse button before the next call. If you want to break using the mouse, keep the button down until Macaroni pauses. Resume execution by clicking once on the mouse button.

Macaroni Save MyScript...

This command saves a Macaroni script as a text file with the suffix ".macaroni". It puts up a standard Macintosh SFPut dialog for selecting the folder where the file is to be saved. Any file with the same name in the selected folder will be overwritten.

Macaroni Print MyScript...

This command displays the standard Macintosh print dialog which can be used to print the text of the script.

Macaroni Delete MyScript...

This command deletes a Macaroni script and its data structures. The name of the script will be deleted from the Script menu. If a script is saved before it is deleted then it can be restored later by opening the saved script. It is good practice to protect yourself by saving your script before you delete it.

Macaroni Show Window

This command displays the drawing window into which QuickDraw output is directed.

Macaroni Clear Window

This command clears the drawing window, erasing everything that has been painted into the window.

Macaroni Comments

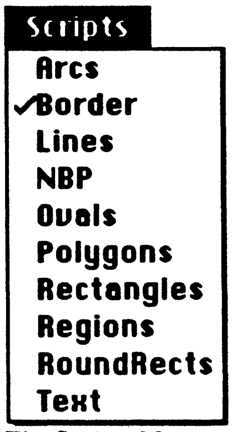
This command displays notes about the release of the version of Macaroni that you are using. It also provides information about how to communicate your comments, bug reports and suggestions regarding Macaroni to the developers.

Macaroni Tutorial

This command takes you to the beginning of a tutorial that shows you how to use Macaroni.

The Scripts Menu

The Scripts menu contains a list of scripts.

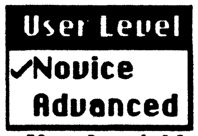


The Scripts Menu

If an item is checked in the Scripts Menu, as in the case of Border, in the picture above, then the Macaroni script with the name of the item that's checked is the current script. To select a script, choose its name in the Scripts Menu.

The User Level Menu

Here is a picture of the User Level menu:



The User Level Menu

When the Novice menu item is selected, Macaroni will prompt you before it executes any call or performs any function like printing a script.

When the Advanced menu item is selected, Macaroni will avoid prompts and it will provide less help when you select a call.

Set the User Level to Novice when you are first getting started with Macaroni. If you find that prompting slows you down, set the User Level to Advanced.

The Help System

Macaroni provides three kinds of help:

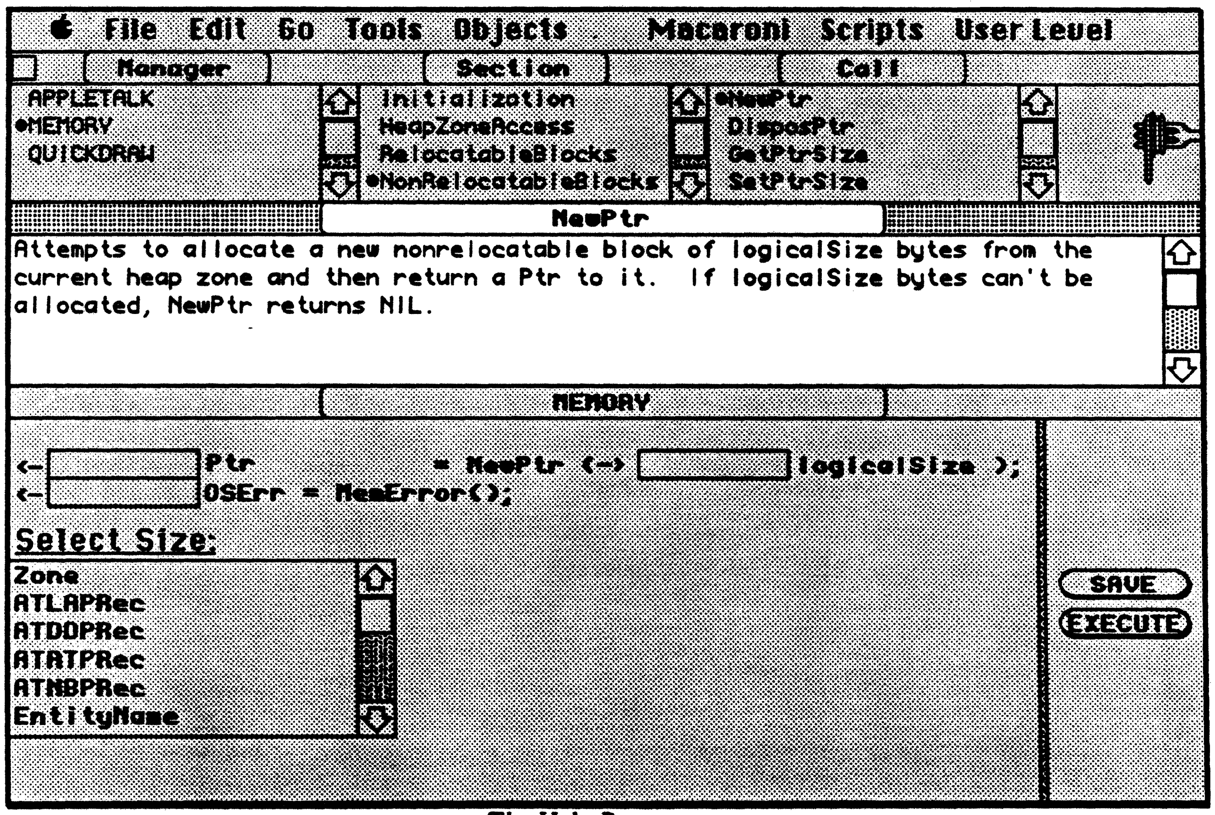
- The Help Pane
- Comments
- Tutorial

Each of these kinds of help is explained below.

The Help Pane

Below the browser is the Help Pane.

Here is a picture showing the Help Pane highlighted for emphasis:



Attempts to allocate a new nonrelocatable block of logicalSize bytes from the current heap zone and then return a Ptr to it. If logicalSize bytes can't be allocated, NewPtr returns NIL.

```
<-[          ]Ptr      = NewPtr (-> [          ]logicalSize );
<-[          ]OSErr = MemError();
```

The Help Pane

The Help Pane contains descriptive help about a call, its input parameters or output results.

In the picture above, the Help Pane contains a description of the *NewPtr* call.

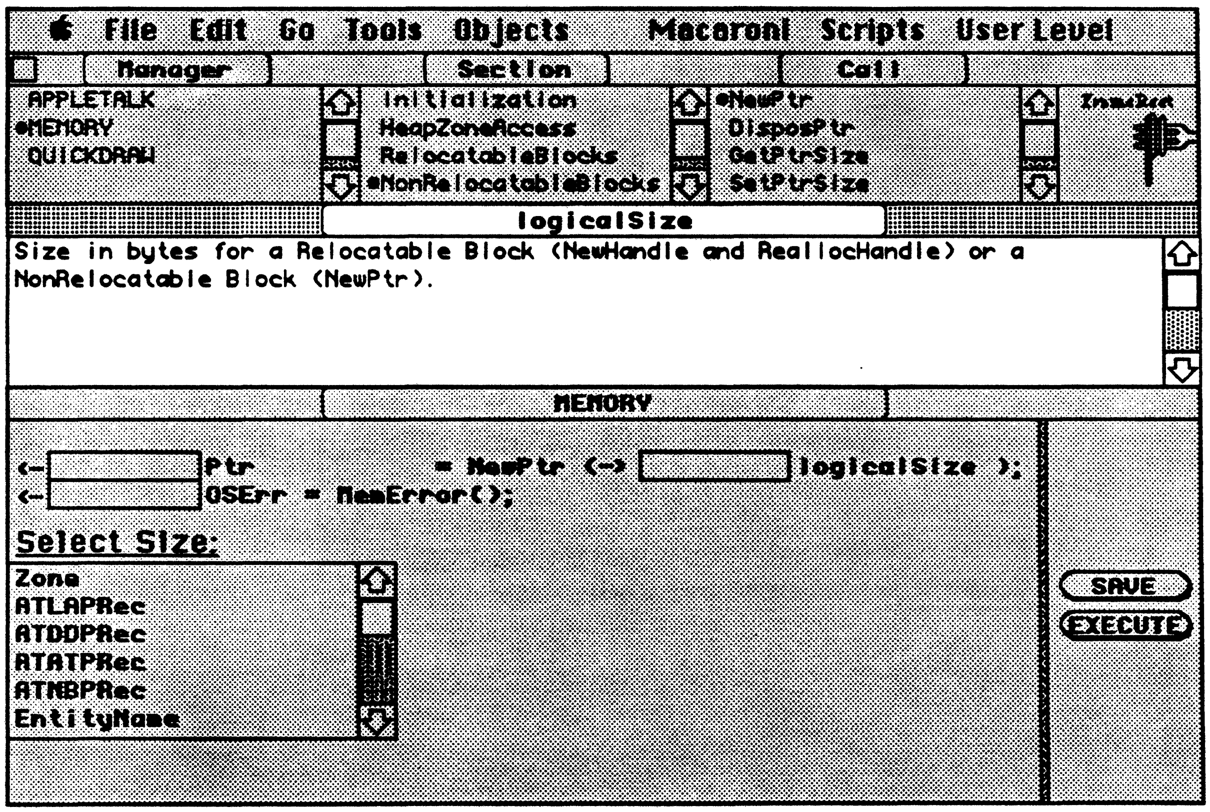
If the User Level is set to Novice, when you first select a call Macaroni puts a description of the call into the Help Pane.

Another way to get help about a call, including a call's parameters or results, is to click in the Toolbox Pane on the name of a call, parameter, or result. Macaroni puts a description of a call, parameter or result into the Help Pane whenever the name of the call, parameter or result is clicked on in the Toolbox Pane.

As an example, to find out more about **logicalSize**, the input parameter to the *NewPtr* call click on the label **logicalSize** in the Toolbox Pane. Below is a picture of the *NewPtr* call with an arrow pointing to the label where you should click:

```
                         ↙
NewPtr (-> [          ]logicalSize );
```

Macaroni will display the following help about the parameter:



Size in bytes for a Relocatable Block (NewHandle and ReallocHandle) or a NonRelocatable Block (NewPtr).

```
<-[          ]Ptr     = NewPtr (-> [          ]logicalSize );
<-[          ]OSErr = MemError();
```

Select Size:

Zone
ATLAPRec
ATDDPRec
ATATPRec
ATNBPRec
EntityName

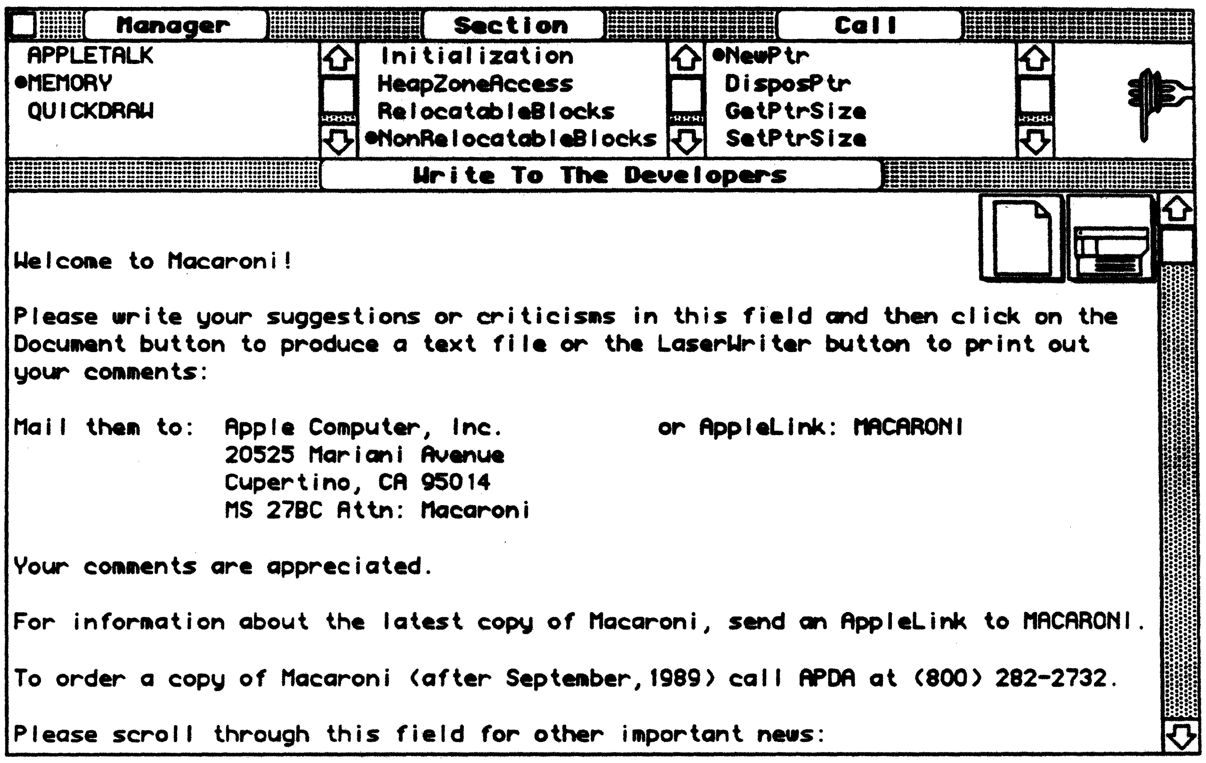
Toolbox Help

Some calls within the Memory Manager contain lists of parameters that have been supplied by the makers of Macaroni as a convenience (e.g. **Select Size** in the picture above). Clicking on an item in such a list will select the item as a parameter rather than display help.

Comments

To read the release notes that accompany Macaroni, select Comments from the Macaroni menu.

Macaroni will display the following message:



```
Welcome to Macaroni!

Please write your suggestions or criticisms in this field and then click on the
Document button to produce a text file or the LaserWriter button to print out
your comments:

Mail them to:  Apple Computer, Inc.          or AppleLink: MACARONI
               20525 Mariani Avenue
               Cupertino, CA 95014
               MS 27BC Attn: Macaroni

Your comments are appreciated.

For information about the latest copy of Macaroni, send an AppleLink to MACARONI.

To order a copy of Macaroni (after September,1989) call APDA at (800) 282-2732.

Please scroll through this field for other important news:
```

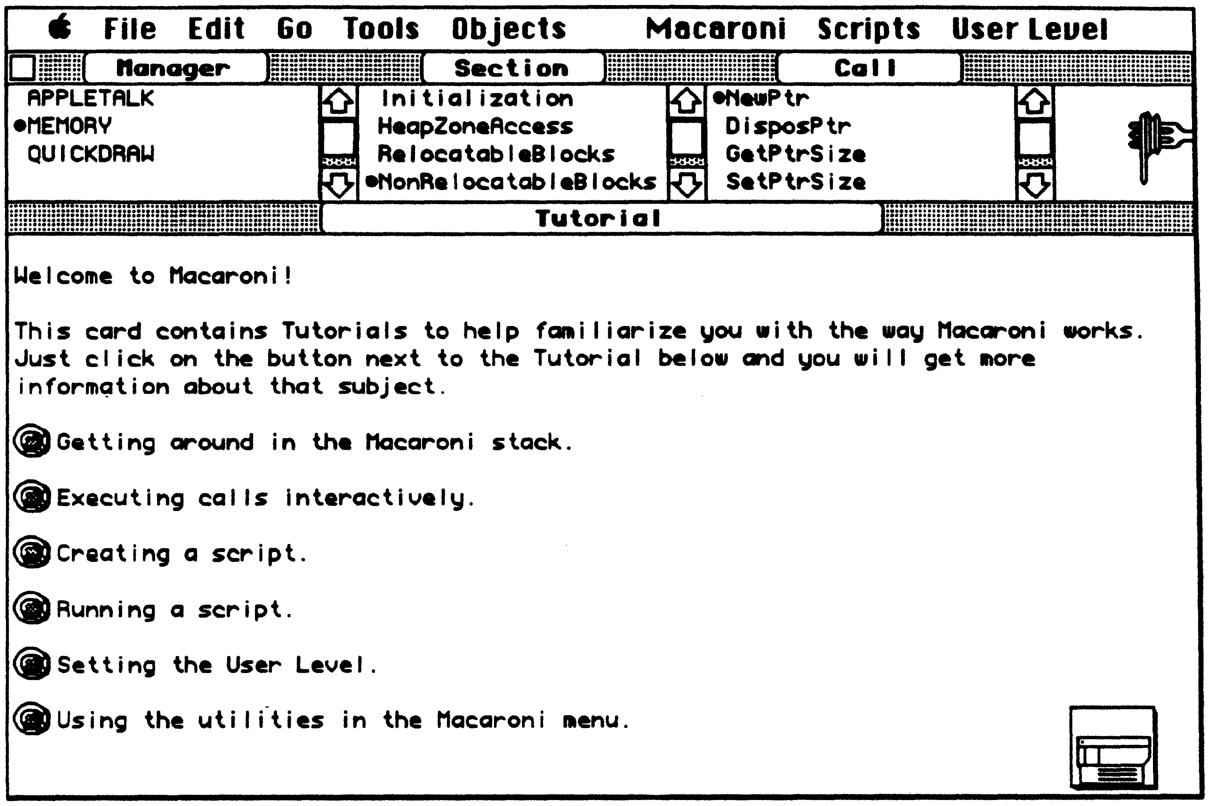
Comments

The Comments also explain how to reach the Developers of Macaroni to communicate suggestions and bug reports.

The Tutorial

To find out more about how to use Macaroni, select Tutorial from the Macaroni menu.

Macaroni will display the following invitation to a tutorial:



Welcome to Macaroni!

This card contains Tutorials to help familiarize you with the way Macaroni works. Just click on the button next to the Tutorial below and you will get more information about that subject.

Getting around in the Macaroni stack.

Executing calls interactively.

Creating a script.

Running a script.

Setting the User Level.

Using the utilities in the Macaroni menu.

The Tutorial

Techniques

Earlier sections of this manual presented the ingredients of the Macaroni interface. The next several sections of this manual present the following techniques for using the interface:

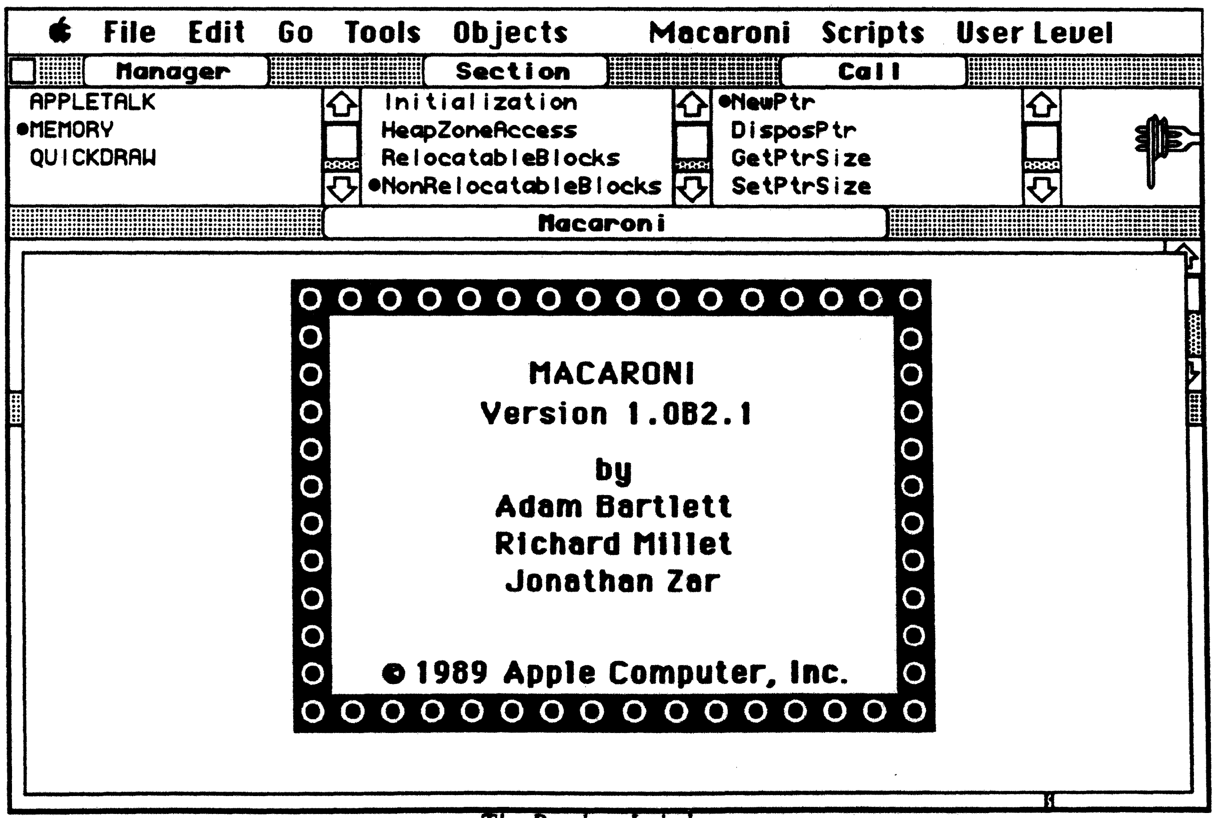
•Launching Macaroni
•Executing A Call
•Scripting.

Getting Started

Here is how to get to a point where you can start cooking with Macaroni.

First, install Macaroni by copying it to your hard disk into a folder which you dedicate to Macaroni. Give the folder any name that you like. When you are ready to run Macaroni, launch it from your hard disk, *not* your floppy drive.

Upon launch, Macaroni displays a product label:

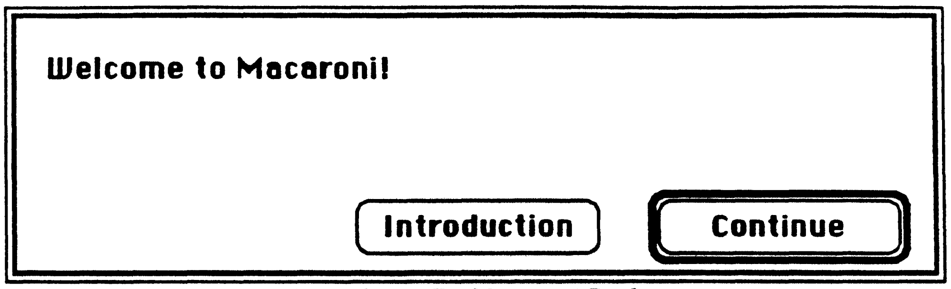


MACARONI
Version 1.0B2.1
by
Adam Bartlett
Richard Millet
Jonathan Zar
© 1989 Apple Computer, Inc.

The Product Label

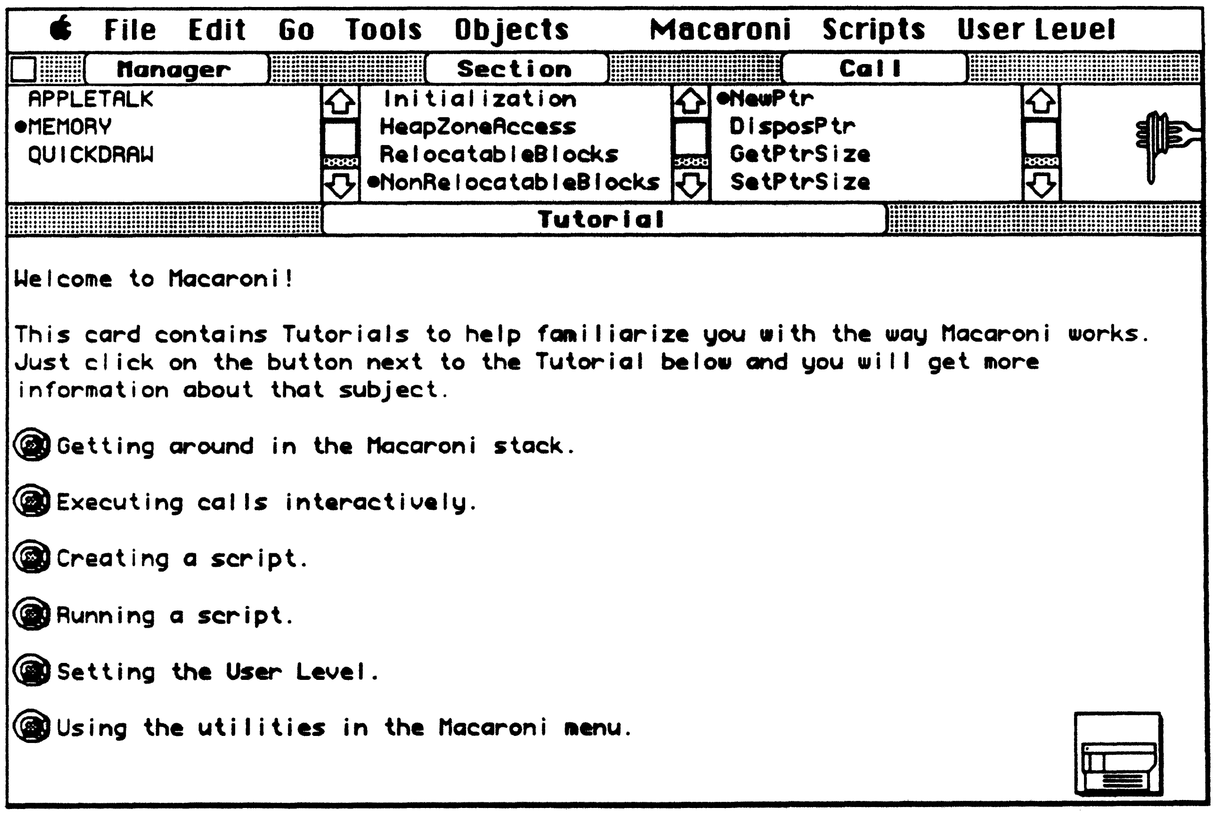
The buttons on the Product Label flash until the mouse is clicked.
Read the product label and then click on the mouse.

After your click, Macaroni displays the following dialog:



The Welcome To Macaroni Dialog

If the Introduction button is pressed Macaroni will display the Tutorial:



Welcome to Macaroni!

This card contains Tutorials to help familiarize you with the way Macaroni works.
Just click on the button next to the Tutorial below and you will get more
information about that subject.

Getting around in the Macaroni stack.

Executing calls interactively.

Creating a script.

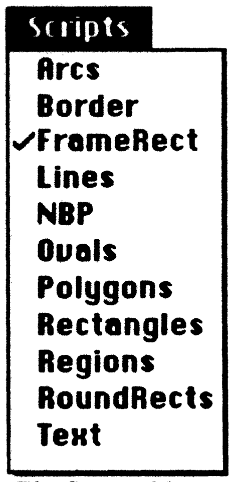
Running a script.

Setting the User Level.

Using the utilities in the Macaroni menu.

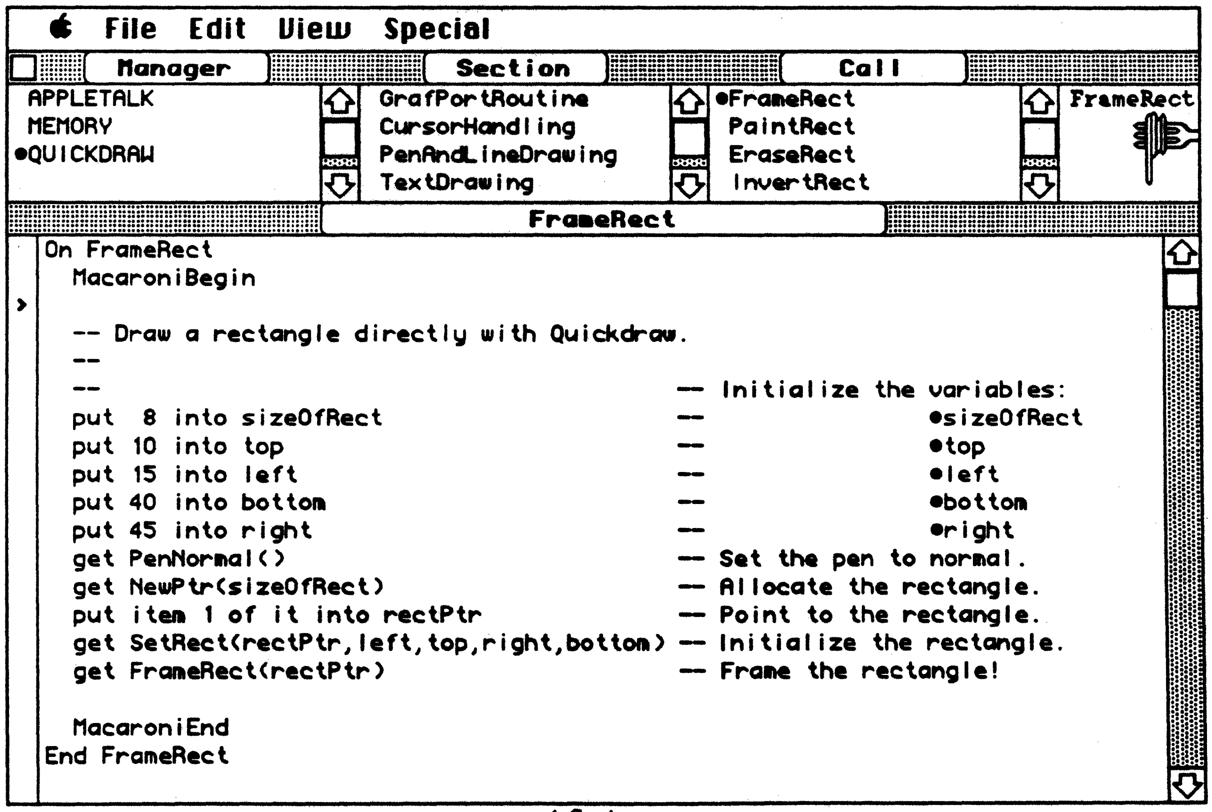
The Tutorial

If the Continue button is pressed in the *Welcome To Macaroni* dialog and if a script is checked in the Scripts menu, then



The Scripts Menu With A Script Checked

Macaroni will display the script:



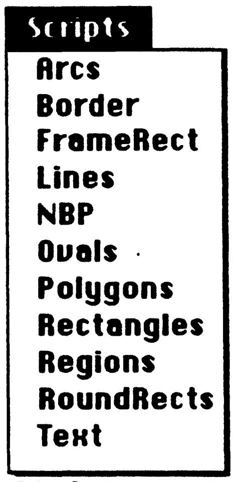
```
On FrameRect
  MacaroniBegin

  -- Draw a rectangle directly with Quickdraw.
  --
  --                                      -- Initialize the variables:
  put  8 into sizeOfRect                  --                  ●sizeOfRect
  put 10 into top                         --                  ●top
  put 15 into left                        --                  ●left
  put 40 into bottom                      --                  ●bottom
  put 45 into right                       --                  ●right
  get PenNormal()                         -- Set the pen to normal.
  get NewPtr(sizeOfRect)                  -- Allocate the rectangle.
  put item 1 of it into rectPtr           -- Point to the rectangle.
  get SetRect(rectPtr,left,top,right,bottom) -- Initialize the rectangle.
  get FrameRect(rectPtr)                  -- Frame the rectangle!

  MacaroniEnd
End FrameRect
```

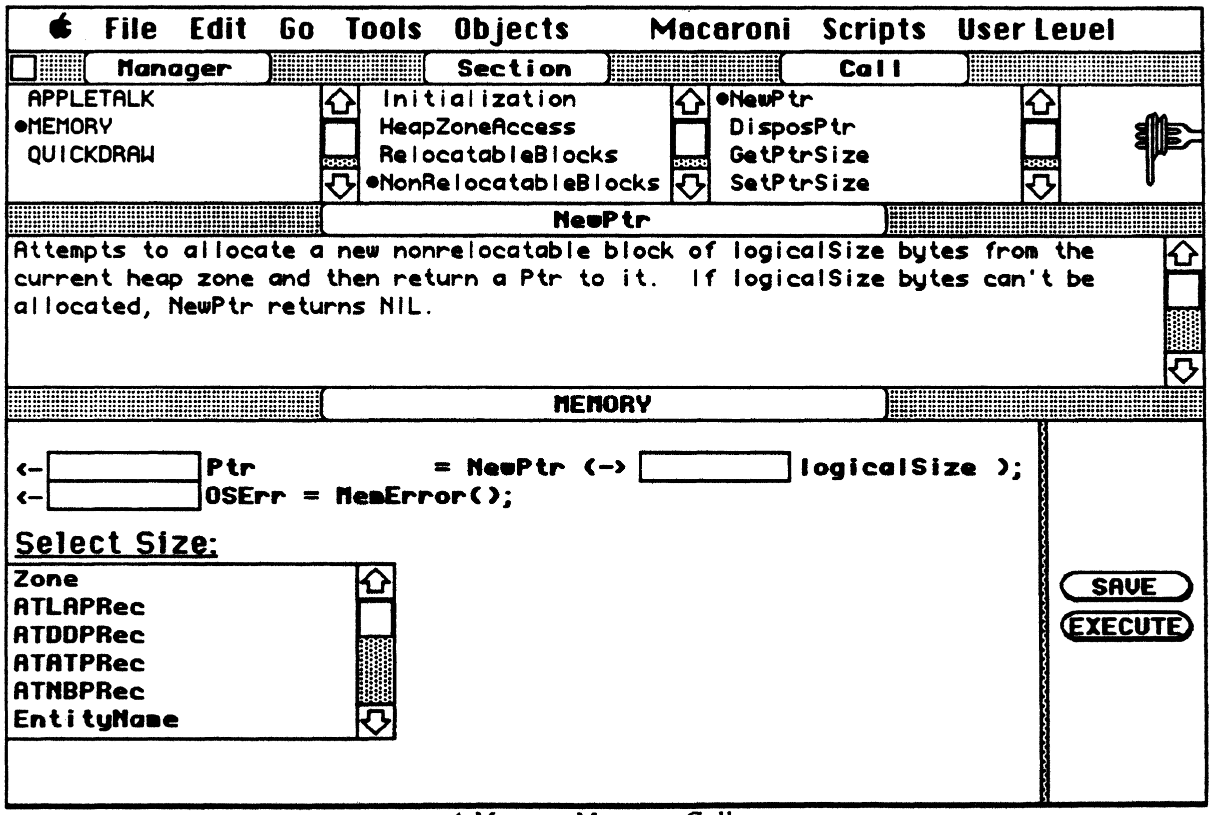
A Script

If the Continue button is pressed in the *Welcome To Macaroni* dialog and if a script is *not* checked in the Scripts menu, then



The Scripts Menu
With No Script Checked

Macaroni will display a call:



Attempts to allocate a new nonrelocatable block of logicalSize bytes from the current heap zone and then return a Ptr to it. If logicalSize bytes can't be allocated, NewPtr returns NIL.

```
<-[          ]Ptr      = NewPtr (->) [          ] logicalSize );
<-[          ]OSErr = MemError();
```

A Memory Manager Call

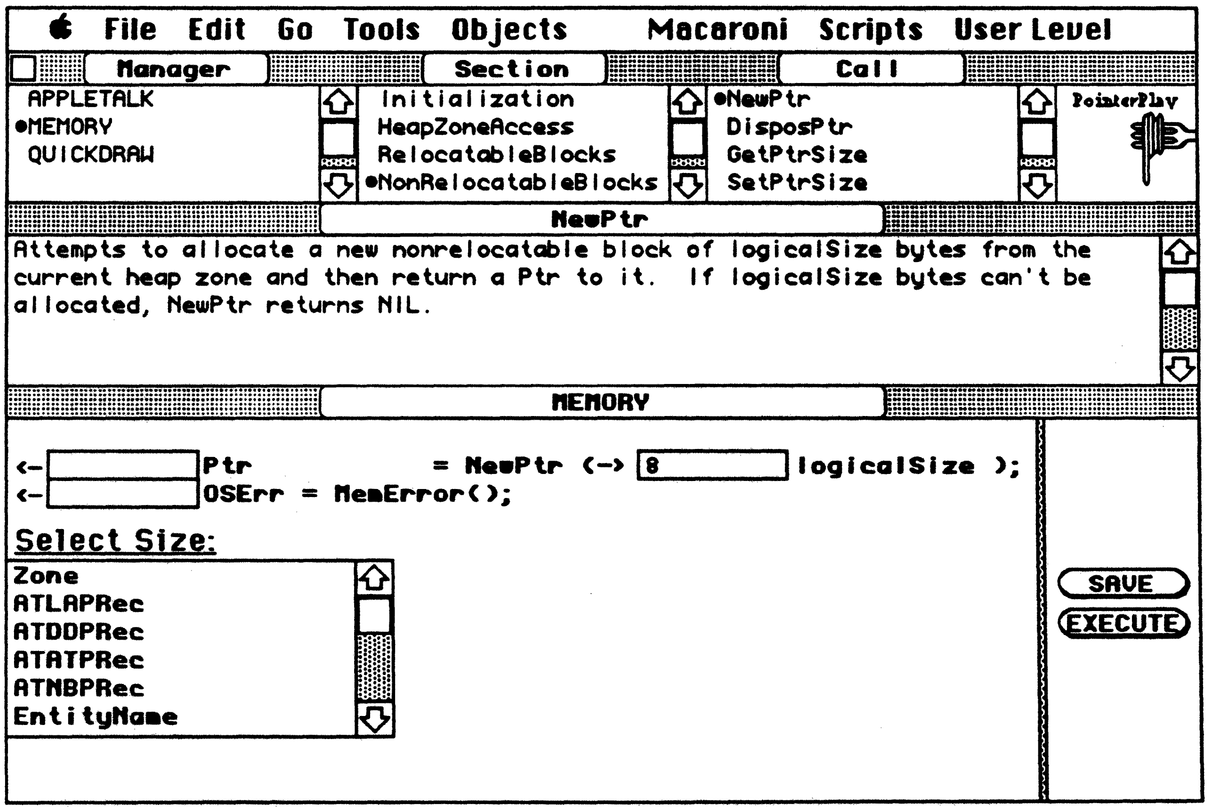
Executing A Call

Macaroni lets you interactively execute a call so that you can explore the Toolbox. To execute a call interactively, fill in the blanks for the input parameters to the Call (in the Toolbox Pane) and press the Execute button (in the Control Pane).

Here is an example taken from the *PointerPlay* tutorial that allocates memory for a rectangle using the *NewPtr* call.

Type the number 8 into the input parameter for the *NewPtr* call (the box labeled **logicalSize**) and click on the **EXECUTE** button.

Here is a picture of the *NewPtr* call after the **logicalSize** field has been filled in and before the **EXECUTE** button has been pressed:

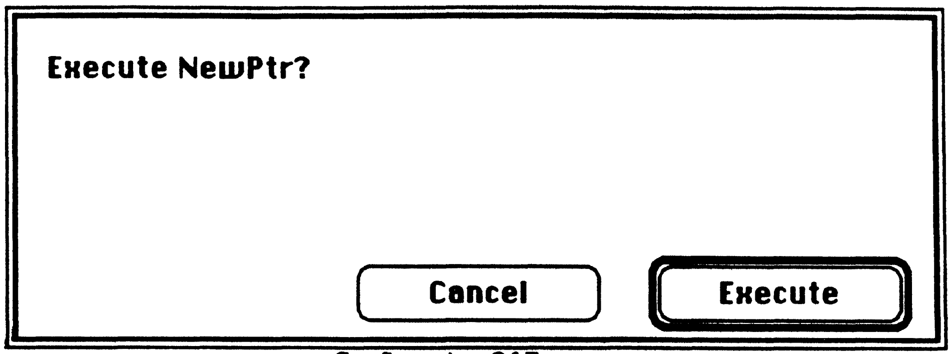


Attempts to allocate a new nonrelocatable block of logicalSize bytes from the current heap zone and then return a Ptr to it. If logicalSize bytes can't be allocated, NewPtr returns NIL.

```
<-[        ]Ptr          = NewPtr (-> [8        ] logicalSize );
<-[        ]OSErr = MemError();
```

The NewPtr Call—Ready To Execute

Once the EXECUTE button is pressed, Macaroni will ordinarily ask for confirmation:

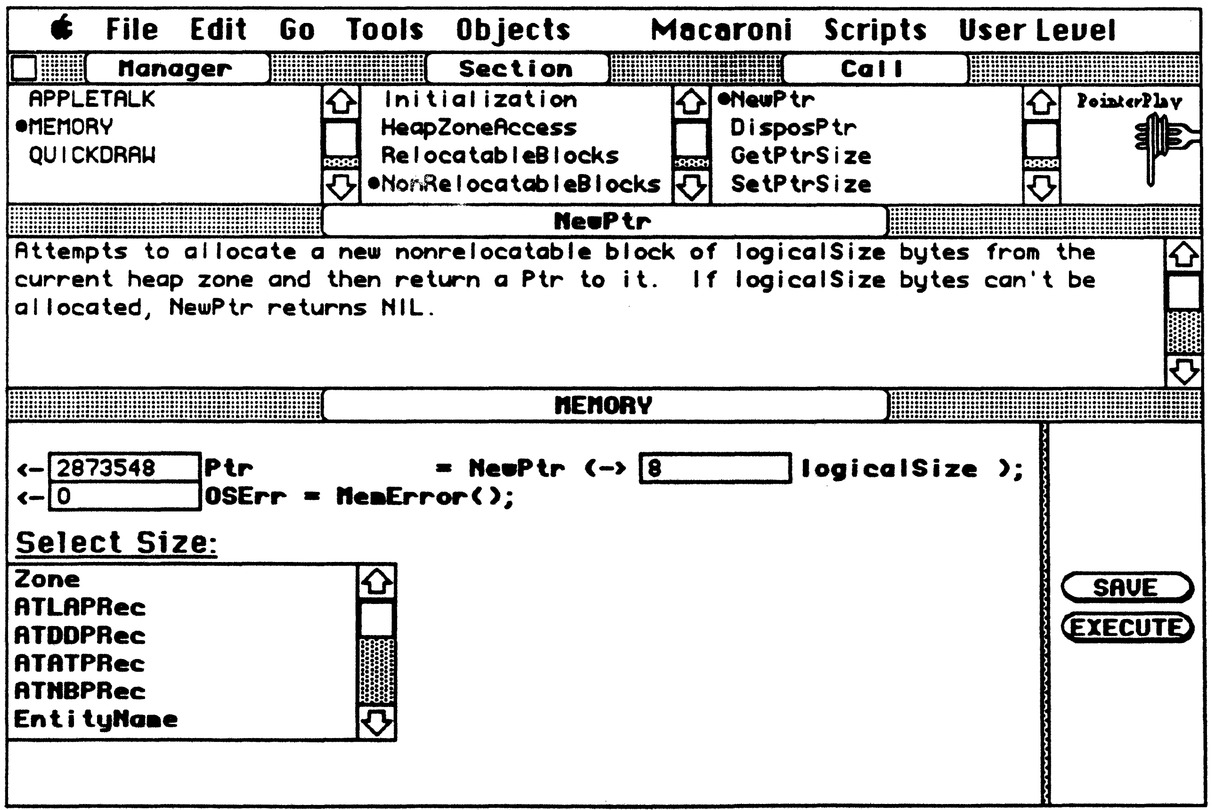

Execute NewPtr?

Confirmation Of Execute

By setting the User Level to Advanced you can make Macaroni skip confirmation.

Once you press the Execute button in the confirmation dialog Macaroni will execute the call. (If you execute *NewPtr* on your machine you will probably see a different number return for the Ptr then is shown in the following picture.)

Here is a picture of the *NewPtr* call immediately after it has been executed:



Attempts to allocate a new nonrelocatable block of logicalSize bytes from the current heap zone and then return a Ptr to it. If logicalSize bytes can't be allocated, NewPtr returns NIL.

```
<- 2873548 Ptr     = NewPtr (-> 8 logicalSize );
<- 0       OSErr = MemError();
```

The NewPtr Call—After Execution

Scripting

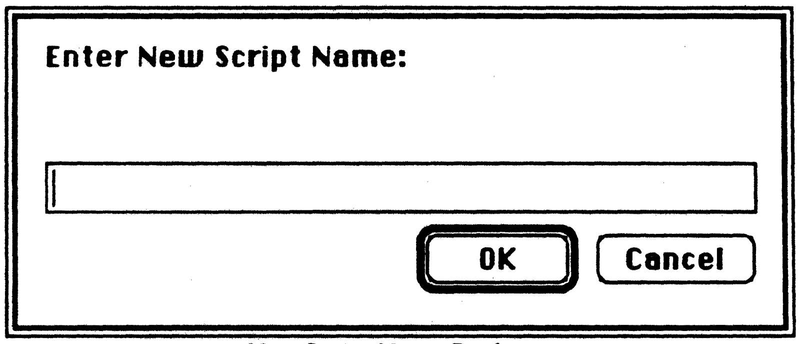
Macaroni supports scripting. A Macaroni script is a named collection of toolbox calls written in HyperTalk. The next several sections of this manual explain how to create a new script, use an existing script, and program a script in HyperTalk.

To learn about HyperTalk, consult HyperCard's online help system, any of the books on HyperTalk in your local bookstore or the following Apple manual published by Addison-Wesley Publishing Company, Inc: *HyperCard Script Language Guide, The HyperTalk Language.*
For information on how to order this manual from APDA call:
(800) 282-2732.

Creating an Empty Script

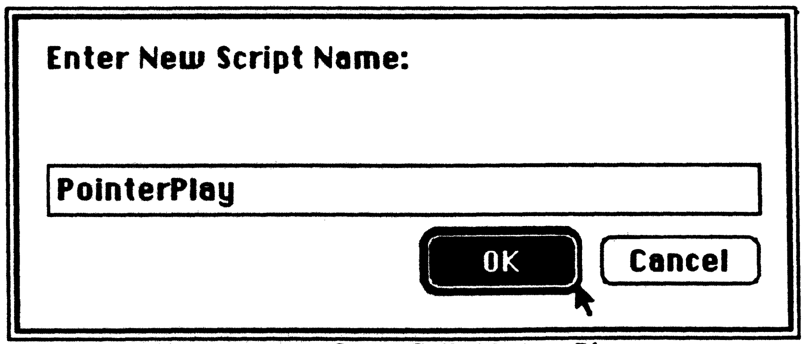
To create a new script, select the menu item **New Script...** from the Macaroni menu.

Macaroni will display a dialog that calls for the name of the new script:



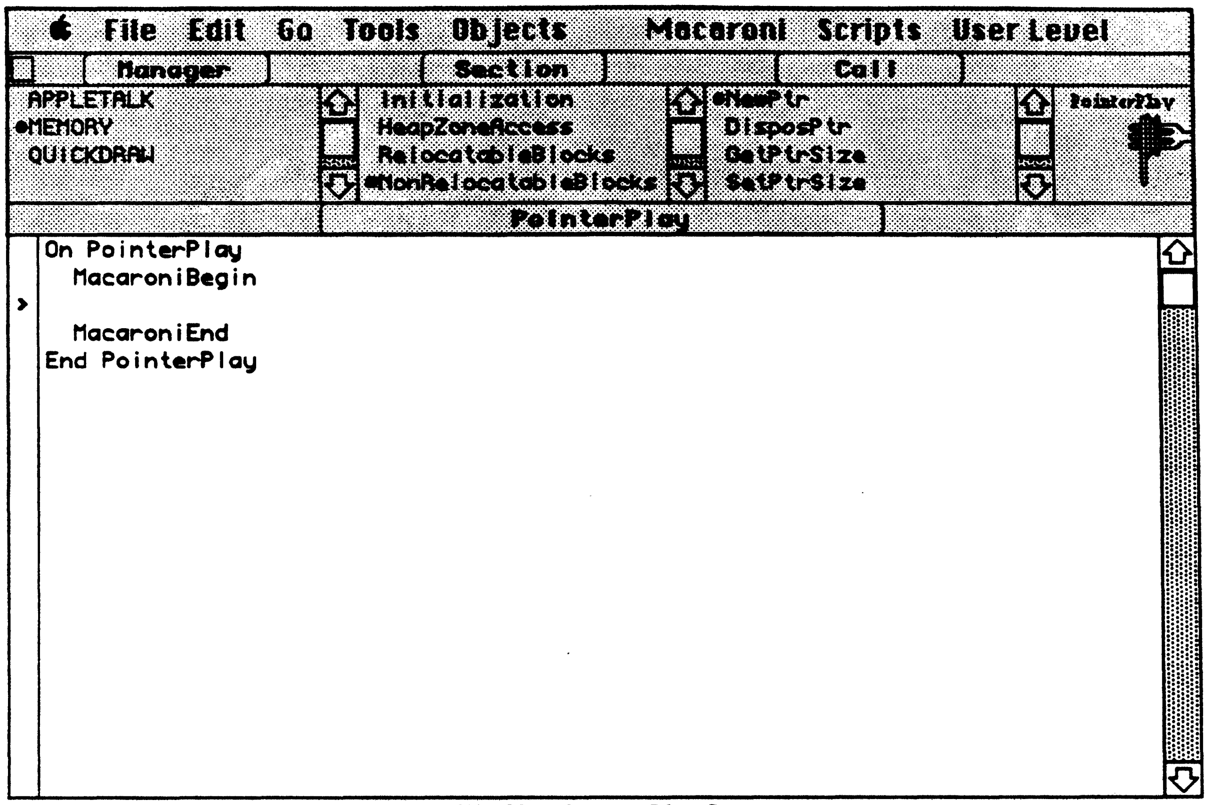
New Script Name Dialog

Respond to the dialog by entering the name "PointerPlay" and clicking on the "OK" button:



Naming A New Script Called PointerPlay

Macaroni will add the name *PointerPlay* to the Scripts menu and create a new script card called *PointerPlay* :



```
On PointerPlay
  MacaroniBegin
>
  MacaroniEnd
End PointerPlay
```

The New PointerPlay Script

Macaroni creates the shell of an otherwise empty script and puts this script into the text of the script card.

Here is a close-up view of the upper-left corner of the script:

```
| |On PointerPlay
| |  MacaroniBegin
|>|
| |  MacaroniEnd
| |End PointerPlay
```

Notice the vertical bars to the left of the text of the script:

```
| |
|>|
| |
```

These bars hold the script insertion point, a caret (>), which marks where Macaroni is to add code to the script.

Using a Script

To save *PointerPlay*, select the Save PointerPlay ... command from the Macaroni menu and respond to the dialog by naming the script and choosing a folder into which the script should be saved. Macaroni will respond by saving a copy of the script as a text file with the extension ".macaroni".

To open *PointerPlay* after it has been saved, select the Open PointerPlay ... command from the Macaroni menu. Macaroni will respond by creating a new script card and adding the name "PointerPlay" to the Scripts menu. The file on disk named "PointerPlay.macaroni" will remain unchanged.

To print the contents of *PointerPlay*, select the Print PointerPlay... command from the Macaroni menu. Macaroni will display the standard Macintosh print dialog to let you print the text of the *PointerPlay*.

To delete *PointerPlay*, select the Delete PointerPlay... command from the Macaroni menu. Macaroni will respond by deleting the script card on which *PointerPlay* is recorded and by deleting the name "PointerPlay" from the Scripts menu.

To run *PointerPlay*, select the Run PointerPlay... command from the Macaroni menu. Macaroni will display the Run Dialog to let you set options and run the *PointerPlay* script.

Programming a Script

Recall the text of the *PointerPlay* script:

```
on PointerPlay
   MacaroniBegin

   MacaroniEnd
end PointerPlay
```

This code is automatically generated by Macaroni when a new script is created. By itself it merely prepares to do something and then leaves. It is a model (or "shell") for adding other code.

There are two ways to add code to a script. One method is to save a call from the Toolbox Pane. The other method is to enter a call using the script editor

Any call shown in Macaroni can be saved into a script by clicking on the SAVE button in the Control Pane that is adjacent to the call and confirming the dialog. (Confirm the dialog by clicking on the **SAVE** button).

Another method for entering code into a script is to use the HyperTalk Script Editor. Open the Editor by clicking on the text of the script.

The text of every script must obey seven rules:

- Name A Handler After The Script
- Script Between MacaroniBegin and MacaroniEnd
- Call The Toolbox Using HyperTalk Functions
- Use Pascal Names Unless They Conflict With HyperTalk
- Extract Results As HyperTalk Items
- Avoid Globals
- Accept The Consequences Of Calling The Toolbox

Each of these rules is explained below.

Rule 1—Name A Handler After The Script

Every Macaroni script must contain one handler that is named after the script. The handler must be a procedure that does not take arguments.

When a Macaroni script is run, Macaroni sends the name of the script as a message to a field on the card where the handlers that constitute the script are stored. Sending the message causes HyperCard to attempt to give control to a handler in the field that is named after the script. Macaroni expects the handler to be a procedure and it does not pass any arguments.

Rule 2—Script Between *MacaroniBegin* and *MacaroniEnd*

The second line of the first handler to get control must be *MacaroniBegin* and the second-to-last line of the handler must be *MacaroniEnd.*

Here is an example from the text of a script called *sampleScript:*

```
On sampleScript
     MacaroniBegin
     DrawRectangle(NewRectangle(10,20,40,60))
     MacaroniEnd
End sampleScript
```

MacaroniBegin and MacaroniEnd are handlers defined by Macaroni that handle initialization and cleanup. *MacaroniBegin* must be executed before the rest of a Macaroni script is run so that Macaroni can prepare to run a script. *MacaroniEnd* must be executed after the rest of a Macaroni script has run so that Macaroni can cleanup memory and write out a log file.

Rule 3—Call The Toolbox Using HyperTalk Functions

In Macaroni every Toolbox call must be in the form of a HyperTalk function even if the Toolbox does not return a value and could be called from Pascal as a procedure.

Below is the text of a script called *MyScript* in which every Toolbox call is highlighted in italics for emphasis:

```
On MyScript
  MacaroniBegin
  --Draw a rectangle object--
  DrawRectangle(NewRectangle(10,20,40,60))
  MacaroniEnd
End MyScript

function NewRectangle top,left,bottom,right
  --Create a rectangle object
  put 8 into sizeofRect
  put NewPtr(sizeofRect) into result
  put item one of result into RectPtr
  put item two of result into OSErr
  put sizeofRect & RETURN ¬
  &  top      & RETURN ¬
  &  left     & RETURN ¬
  &  bottom   & RETURN ¬
  &  right    & RETURN ¬
  &  rectPtr   into MyRect
  return MyRect
end NewRectangle

on DrawRectangle MyRect
  --Draw a rectangle object
  put line 1 of MyRect into sizeofRect
  put line 2 of MyRect into top
  put line 3 of MyRect into left
  put line 4 of MyRect into bottom
  put line 5 of MyRect into right
  put line 6 of MyRect into rectPtr
  get PenNormal()
  get SetRect(rectPtr,left,top,right,bottom)
  get FrameRect(rectPtr)
end DrawRectangle
```

Note that each Toolbox call is a HyperTalk function:

```
        put NewPtr(sizeofRect) into result
        get PenNormal()
        get SetRect(rectPtr,left,top,right,bottom)
        get FrameRect(rectPtr)
```

Rule 4—Use Pascal Names Unless They Conflict With HyperTalk

Toolbox calls have Pascal names unless their name as defined in the Pascal interface within Inside Macintosh conflicts with the name of a reserved word in HyperTalk.

When a script is run each toolbox call is sent as a message to a handler that calls an XCMD for the manager that owns the call. This scheme only works if calls are interpreted by HyperTalk as handlers. Reserved words are not interpreted as handlers and calls that have the same name as a reserved word must use an alias.

In the beta version of Macaroni the only conflict is with the call *Line*. The alias for this call is *QDLine*.

Rule 5—Extract Results As HyperTalk Items

Results from Toolbox calls are returned as a list whose items are separated by commas.

Consider the following allocation of a rectangle from the *NewRectangle* call:

```
put NewPtr(sizeofRect) into result
put item one of result into RectPtr
put item two of result into OSErr
```

The first item returned from the call is RectPtr. The second item that is returned is OSErr.

In the case of the *NewPtr* call, Macaroni concatenates the results of the call with the results of a call to *MemError*. Whenever you make a memory manager call in Macaroni, the second item of the result of every call is OSErr.

Consider the following extraction of data from an AppleTalk NPB Record:

```
put GetATNBPRec(MEMORY1_ABRecHandle) into fields
put item 1  of fields into APPLETALK4_abOpcode
put item 2  of fields into APPLETALK4_abResult
put item 3  of fields into APPLETALK4_abUserRef
put item 4  of fields into APPLETALK4_nbpEntityPtr
put item 5  of fields into APPLETALK4_nbpBufPtr
put item 6  of fields into APPLETALK4_nbpBufSize
put item 7  of fields into APPLETALK4_nbpDataField
put item 8  of fields into APPLETALK4_nbpAddress_aNet
put item 9  of fields into APPLETALK4_nbpAddress_aNode
put item 10 of fields into APPLETALK4_nbpAddress_aSocket
put item 11 of fields into APPLETALK4_nbpRetransmitInfo_retransInterval
put item 12 of fields into APPLETALK4_nbpRetransmitInfo_retransCount
```

In the preceding example, each item is extracted as a distinct item.

Rule 6—Avoid Globals

Most of the globals used by Macaroni begin with the letter "m" (for Macaroni) so avoid this naming convention if you decide to put globals into your scripts.

HyperCard limits the total number of globals so scripts that use too many globals may not run correctly.

Rule 7—Accept The Consequences Of Calling The Toolbox

Scripts won't run if they contain violations of HyperTalk syntax and Toolbox calls won't work correctly unless they are made in the correct order (initialization before use, allocation before access).

Reinitialization of Toolbox managers and other calls that affect the HyperCard runtime environment are likely to abort HyperCard.

Macaroni was designed to let you explore the Toolbox without hiding is dangers.

Before you run Macaroni, back up all files that may be opened when Macaroni is running. This includes all documents that may be opened in different applications under MultiFinder.

Basic Recipes

Rehearsing *PointerPlay*

PointerPlay is a tutorial that demonstrates how to use the Macintosh Toolbox to do the following:

- allocate a rectangle
- set the rectangle's values in memory
- inspect the values
- draw the rectangle
- dispose of the data in memory

Earlier sections of this manual show how to launch Macaroni and get to either the Tutorial or a script card. The balance of this manual assumes that you have correctly set up, launched, and entered Macaroni.

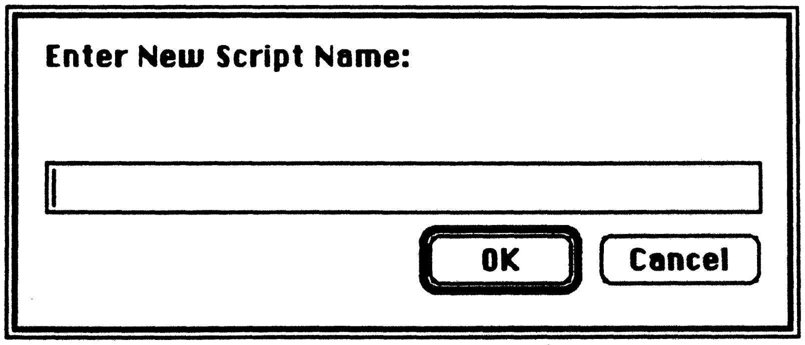
The next section of this manual, "Creating the Script", shows how to create an empty script for *PointerPlay* .

The remaining sections of this manual rehearse the solution to *PointerPlay* call-by-call and then demonstrate how to write a script that uses each of the calls.

Creating the Script

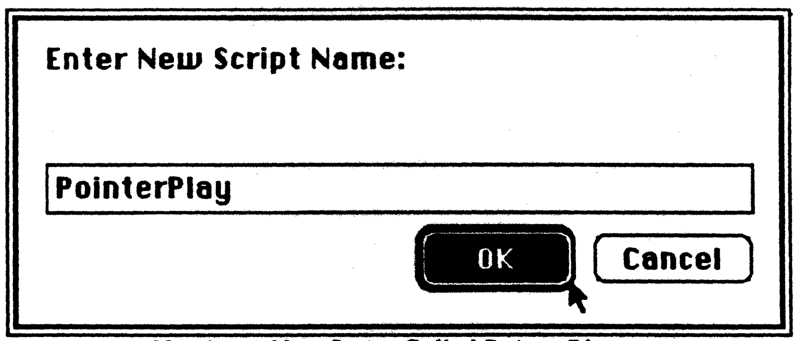
To create a new script for *PointerPlay*, select the menu item **New Script...** from the Macaroni menu.

Macaroni will display a dialog that calls for the name of the new script:



The New Script Name Dialog

Respond to the dialog by entering the name "PointerPlay" and clicking on the "OK" button:

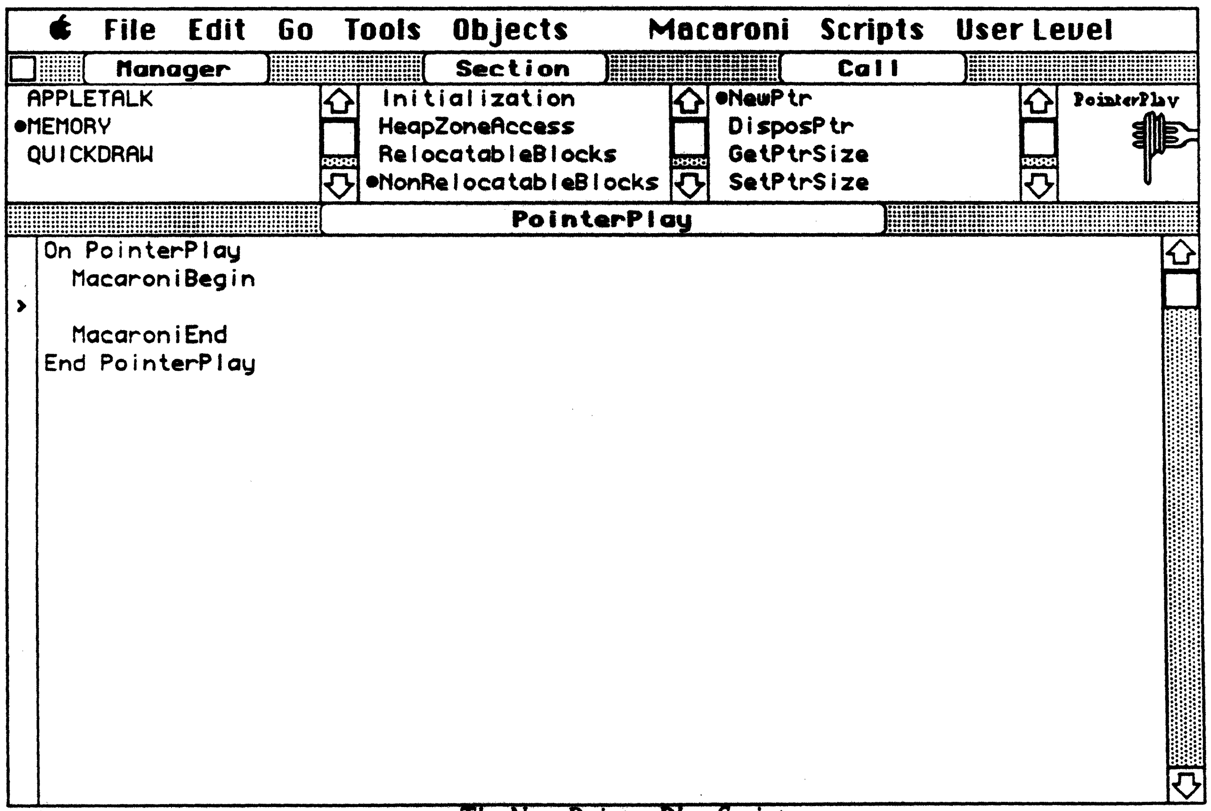


Enter New Script Name:

PointerPlay

Naming A New Script Called PointerPlay

Macaroni will add the name *PointerPlay* to the Scripts menu and create a new script card called *PointerPlay* :



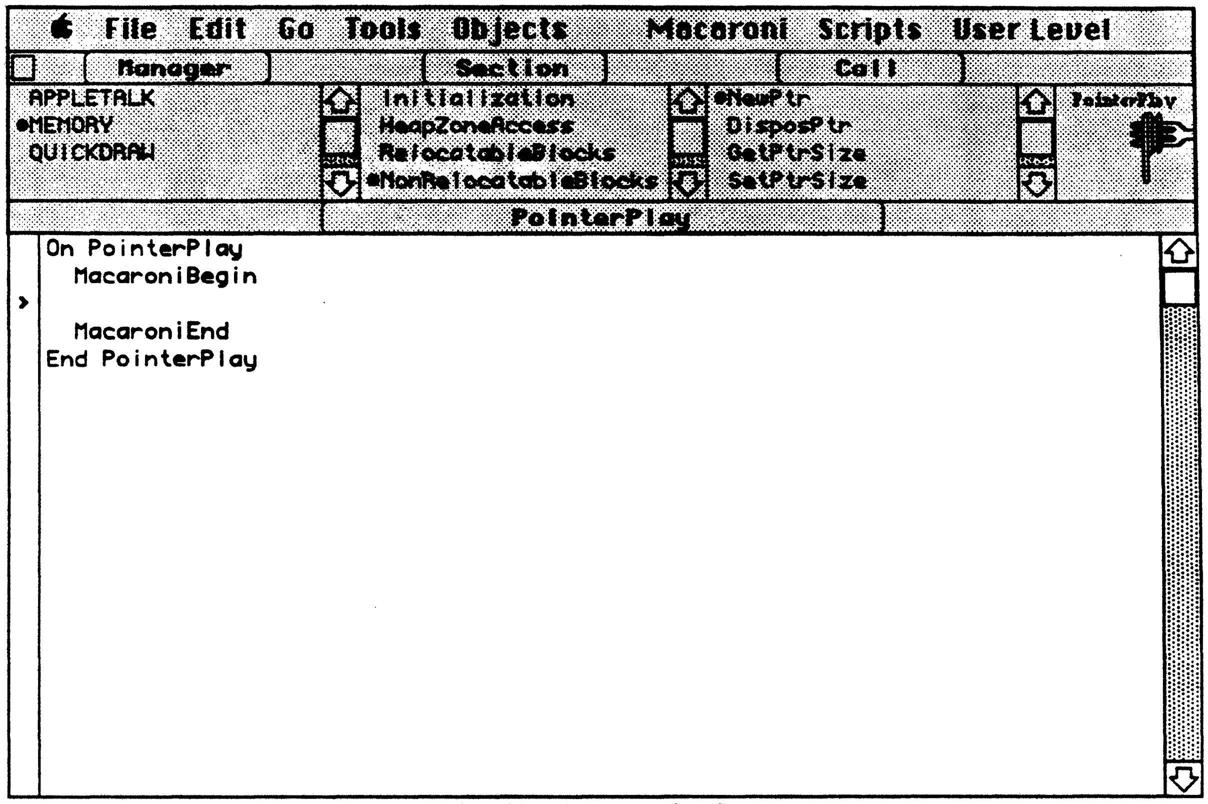
```
On PointerPlay
  MacaroniBegin

  MacaroniEnd
End PointerPlay
```

The New Pointer Play Script

Macaroni creates the shell of an otherwise empty script and puts this script into the text of the script card.

Below is a picture of the script card showing the text of the script highlighted for emphasis:



```
On PointerPlay
  MacaroniBegin
>
  MacaroniEnd
End PointerPlay
```

The New Pointer Play Script

Here is a close-up view of the upper-left corner of the script:

```
| |On PointerPlay
| |  MacaroniBegin
|>|
| |  MacaroniEnd
| |End PointerPlay
```

Notice the vertical bars to the left of the text of the script:

```
| |
|>|
| |
```

These bars hold the script insertion point, a caret (>), which marks where Macaroni is to add code to the script.

Getting a Pointer

The first part of the problem to be solved in this tutorial is the allocation of memory. This is a job for the Memory Manager. The Memory Manger allocates blocks of memory.

To allocate a pointer to a rectangle we need to make a call in the Memory Manager called *NewPtr* and supply the call with the logical size of a rectangle. A rectangle is eight bytes long so its logical size is eight.

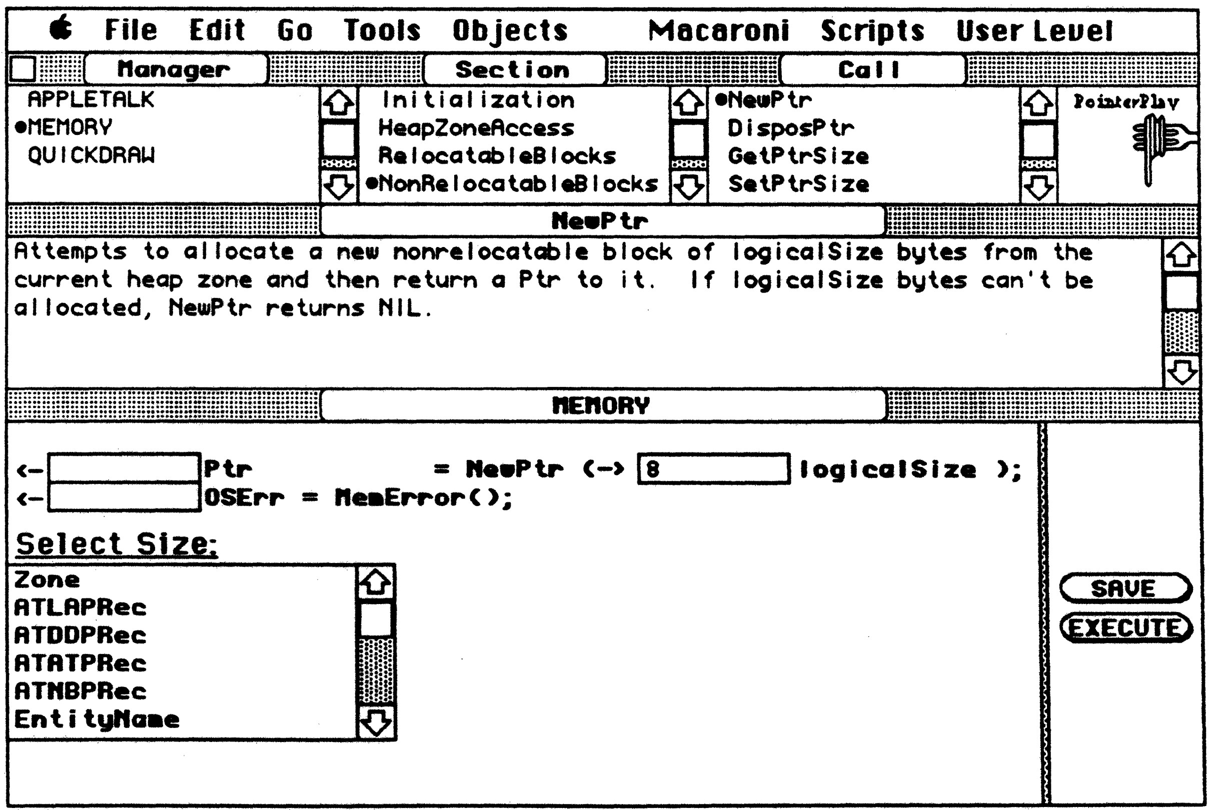
To display the *NewPtr* call do the following:

- Click on the word *Memory* in the Manager Pane of the browser
  wait for Section Pane to update
- Click on the word *NonRelocatableBlocks* in the Section Pane
  wait for the Call Pane to update
- Click on the word *NewPtr* in the Call Pane

Macaroni will display the *NewPtr* call.

To allocate a pointer to a rectangle, type the number 8 into the box labeled **logicalSize** and click on the **EXECUTE** button.

Here is a picture of the NewPtr call after the **logicalSize** field has been filled in and before the **EXECUTE** button has been pressed:

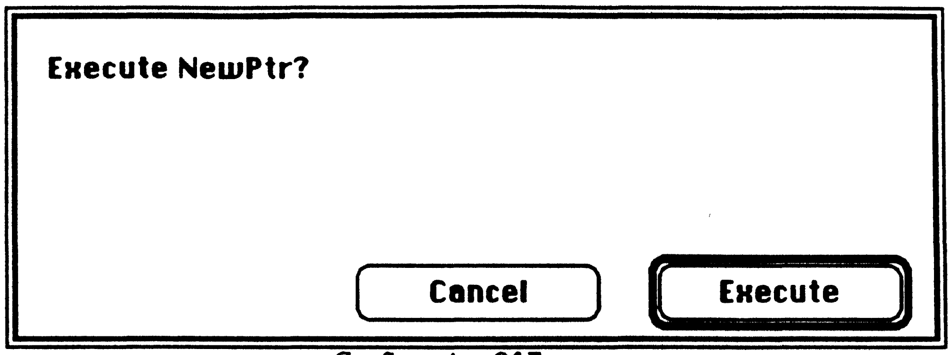


Attempts to allocate a new nonrelocatable block of logicalSize bytes from the current heap zone and then return a Ptr to it. If logicalSize bytes can't be allocated, NewPtr returns NIL.

```
<-[          ]Ptr          = NewPtr (-> [8         ] logicalSize );
<-[          ]OSErr = MemError();
```

The NewPtr Call—Ready To Execute

Once the **EXECUTE** button is pressed, Macaroni will ordinarily ask for confirmation:

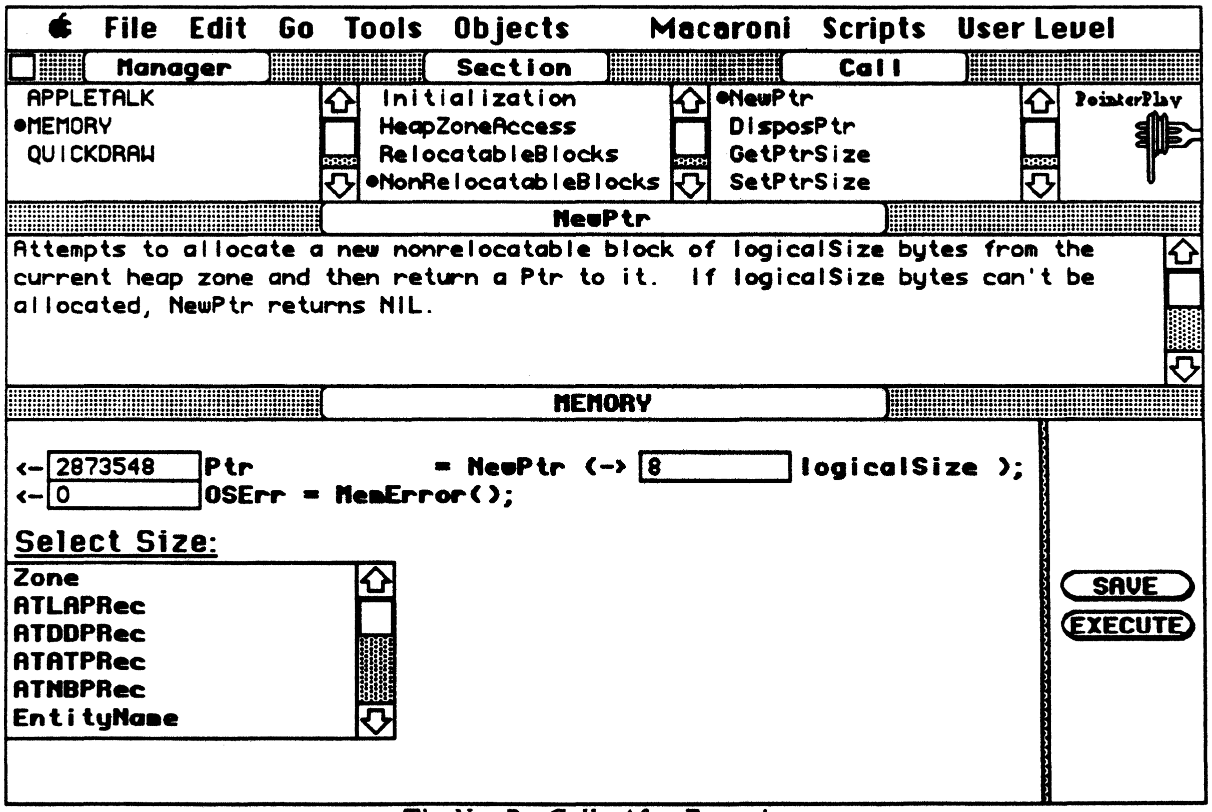


Execute NewPtr?

Confirmation Of Execute

By setting the User Level to Advanced you can make Macaroni skip confirmation. Once you press the Execute button in the confirmation dialog Macaroni will execute the call.

Here is a picture of the *NewPtr* call after it has been executed:



Attempts to allocate a new nonrelocatable block of logicalSize bytes from the current heap zone and then return a Ptr to it. If logicalSize bytes can't be allocated, NewPtr returns NIL.

```
<-[2873548  ]Ptr        = NewPtr (-> [8        ] logicalSize );
<-[0        ]OSErr = MemError();
```

The NewPtr Call—After Execution

Notice that the call returns a number in the field labeled **Ptr**.

If you execute the *NewPtr* call several times you will get a different number each time since every new pointer must point to a new block of memory.

(Similarly, if you execute the *NewPtr* call on your own machine you will probably get a different number then the number shown in the picture above.)

A pointer is valid if it is greater than zero (i.e. greater then **NIL**, a constant equal to zero that means something doesn't exist).

You can make sure that your pointer is valid by checking to see that the **OSErr** field is zero. The **OSErr** field holds the error code returned by the *MemError* function. *MemError* returns zero after a *NewPtr* call if the call is successful. Macaroni always calls *MemError* after every Memory Manager call and puts the value of OsErr returned by the call into the **OsErr** field.

If you have been following this tutorial using Macaroni, you have just allocated a nonrelocatable block of memory. Write down the number of the pointer (you will use it in the next section of this tutorial) and give yourself some applause.

Initializing Memory

The next task is to initialize (i.e. "set" or "write") the memory.

If you followed the tutorial in Macaroni, you will now have a pointer to a block of memory that is eight bytes long. The next task is to set the contents of the block to the values of a rectangle. In the Macintosh, a rectangle data structure is called a *Rect*.

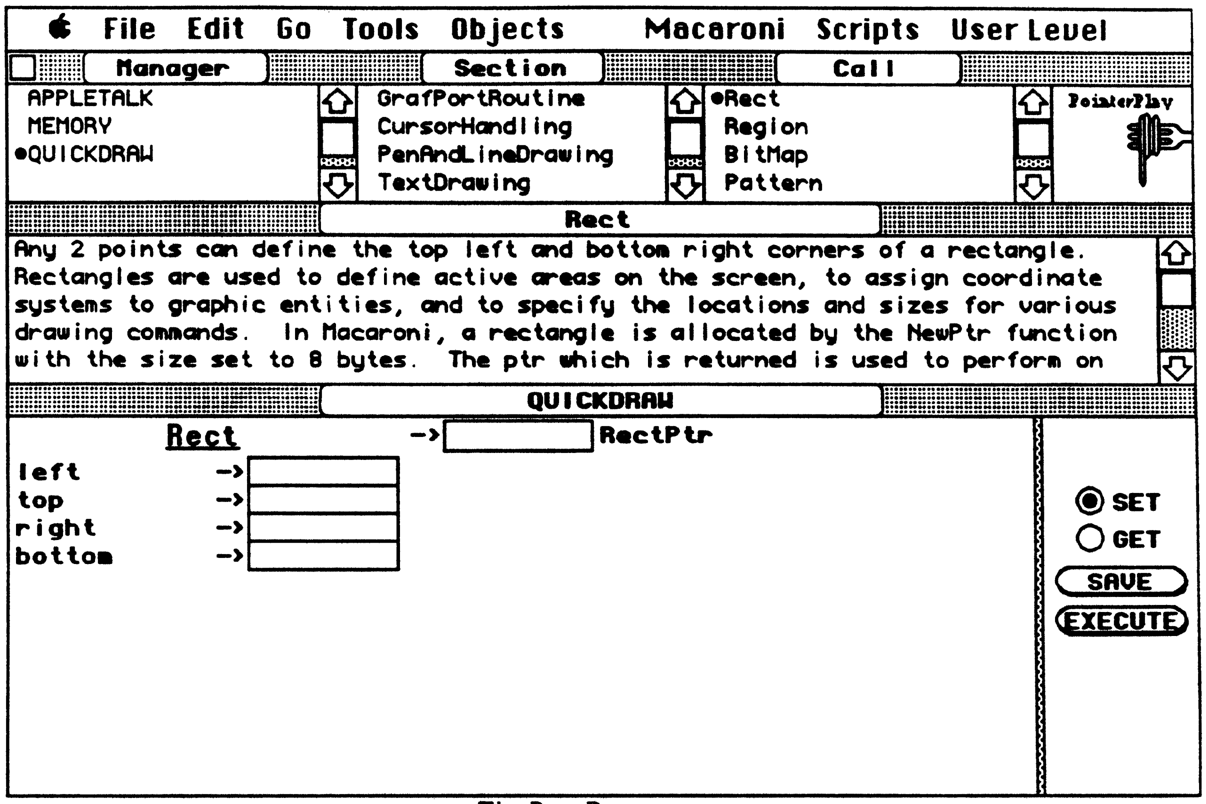
A *Rect* is eight bytes long. It consists of four 16-bit integers that specify the top, left, bottom, and right sides of the rectangle. A *Rect* can also be thought of as two 32-bit points which specify the *Rect*'s top-left and bottom-right corners.

There are two methods for setting the values in a rectangle. One method is to use a Toolbox call, *SetRect*, that resides in the *RectangleCalcs* section of the Quickdraw Manager. The other method is to set the values of the rectangle directly in memory.

In general it is good Macintosh programming practice to use a Toolbox call to manipulate a data structure if such a call is provided. However, for the purpose of demonstrating how to use Macaroni to initialize an arbitrary data structure (i.e. a data structure for which no call like *SetRect* exists), the rectangle in *PointerPlay* will be initialized without using *SetRect*.

- Click on the word *Quickdraw* in the Manager Pane of the browser
  wait for Section Pane to update
- Click on the word *DataTypes* in the Section Pane
  wait for the Call Pane to update
- Click on the word *Rect* in the Call Pane

Macaroni will display the following:



Any 2 points can define the top left and bottom right corners of a rectangle. Rectangles are used to define active areas on the screen, to assign coordinate systems to graphic entities, and to specify the locations and sizes for various drawing commands. In Macaroni, a rectangle is allocated by the NewPtr function with the size set to 8 bytes. The ptr which is returned is used to perform on

Rect		
	->	RectPtr
left	->	
top	->	
right	->	
bottom	->	

Options: [✓ SET] GET

The Rect Datatype

Here's how to initialize a rectangle:

- Insert the value of the pointer into the **RectPtr** field
- Select the **Set** button
- Insert the values for the sides of the rectangle
- Click on the **EXECUTE** button.

The **RectPtr** field is the place to put the **Ptr** to the rectangle that was allocated by the *NewPtr* call. Here is a picture of the **RectPtr**:

-> 2873548 **RectPtr**

Notice that the Control Pane in the picture of the Rect Datatype has a ◉ **SET** button for setting a data structure.

Make sure that the **SET** button is selected. If the **GET** button is selected then click on the **SET** button to change the selection.

For the purposes of this tutorial, specify the size of the rectangle with values as shown below:

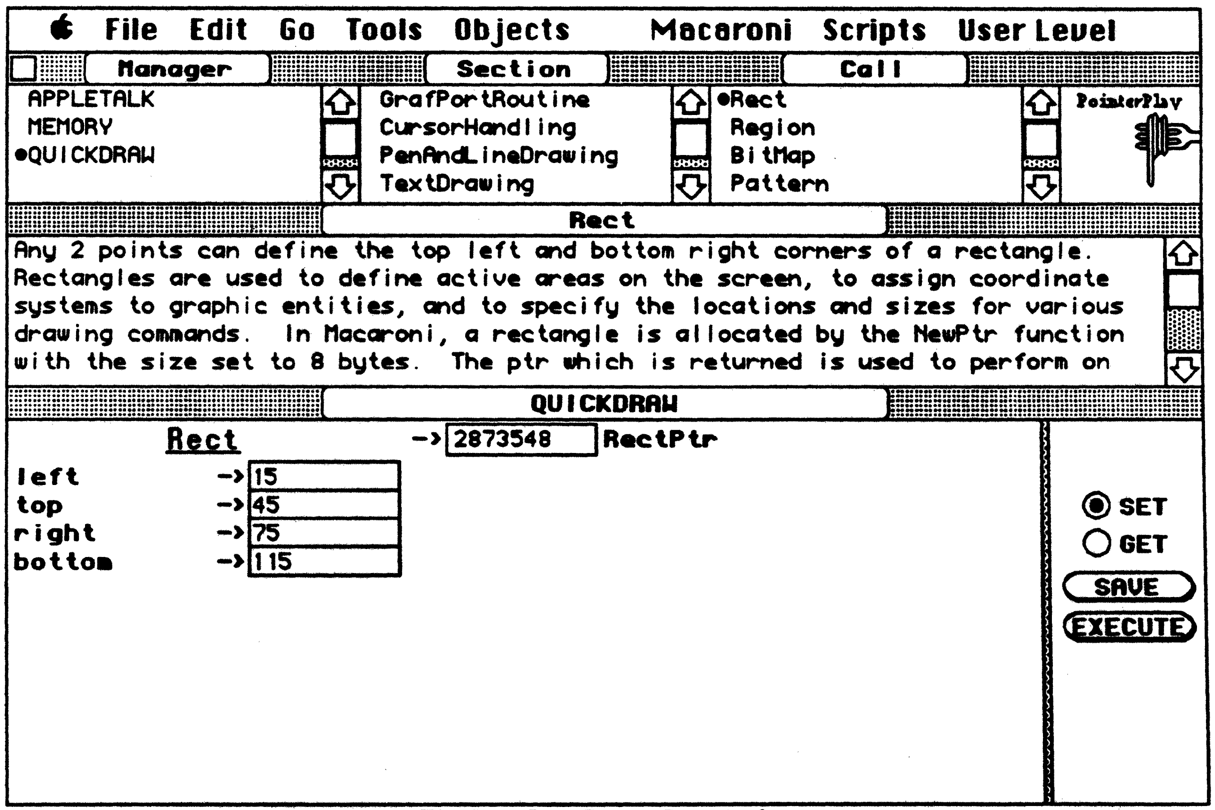
	Rect
left	-> 15
top	-> 45
right	-> 75
bottom	-> 115

Next, click on the **EXECUTE** button and confirm the action in any dialogs that may appear.

Macaroni will initialize the rectangle.

Inspecting Memory

After the the *Rect* has been initialized, Macaroni will look like this:



Any 2 points can define the top left and bottom right corners of a rectangle. Rectangles are used to define active areas on the screen, to assign coordinate systems to graphic entities, and to specify the locations and sizes for various drawing commands. In Macaroni, a rectangle is allocated by the NewPtr function with the size set to 8 bytes. The ptr which is returned is used to perform on

	Rect	
	-> 2873548	RectPtr
left	-> 15	
top	-> 45	
right	-> 75	
bottom	-> 115	

[✓ SET] GET

The Rect DataType—Revisited

To get the values of the rectangle from memory, change the setting of the Control Pane to highlight the **GET** button:

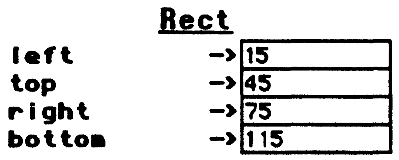
○ SET [✓ ◉ GET]

Then clear the fields for top, left, bottom, and right so that you can easily identify any values that are returned. (To clear a field just highlight and delete.)

Finally, click on the **EXECUTE** button and confirm the **EXECUTE** in any dialogs that may appear.

Macaroni will make the call and return the values of the rectangle. Not surprisingly, the rectangle will have the values that were set for it in the previous section of this manual.

The Rect will look like this:



Rect

```
left     -> 15
top      -> 45
right    -> 75
bottom   -> 115
```

Drawing a Rectangle

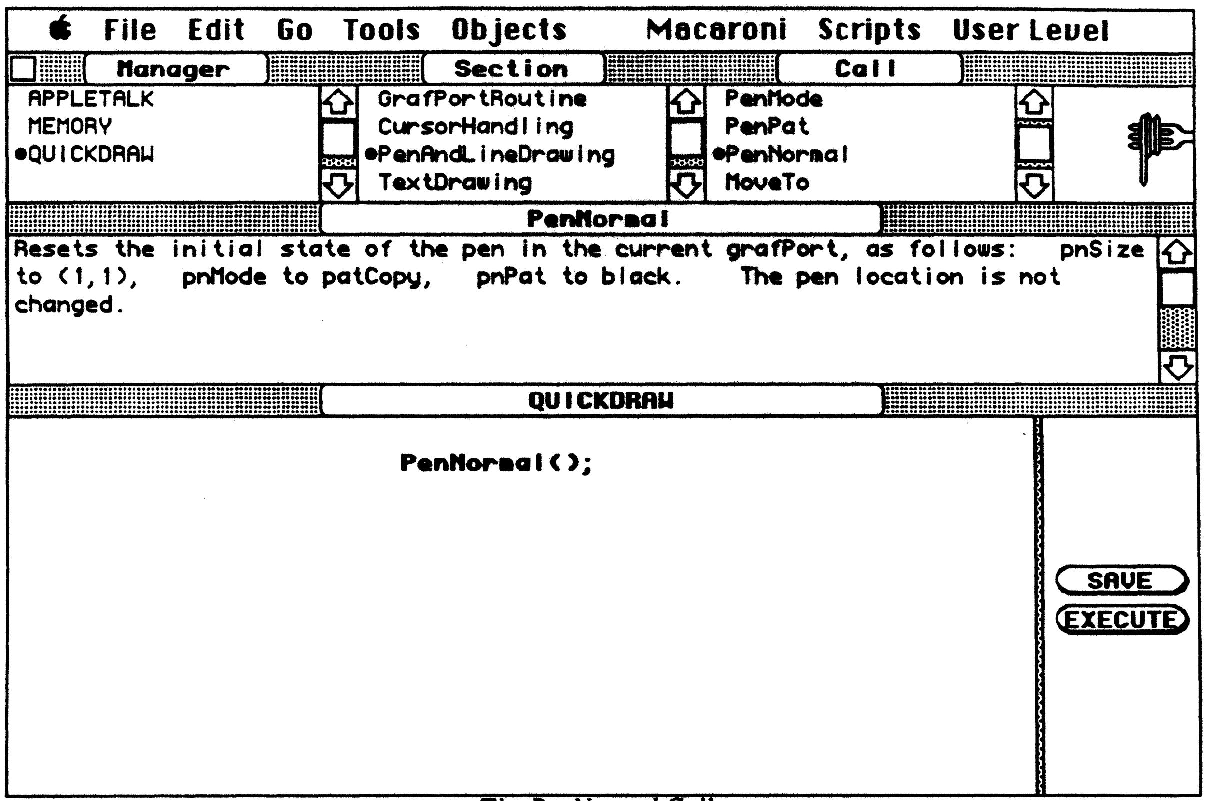
Here are the steps to drawing a black rectangle:

- select and execute the *PenNormal* call
- select the *PaintRect* call, supply a rectangle pointer, and execute the call

The *PenNormal* call resets the Quickdraw drawing environment pen and painting pattern. To select the call do the following:

- Click on the word *Quickdraw* in the Manager Pane of the browser
  wait for Section Pane to update
- Click on the word *PenAndLineDrawing* in the Section Pane
  wait for the Call Pane to update
- Click on the word *PenNormal* in the Call Pane

Macaroni will display the following:



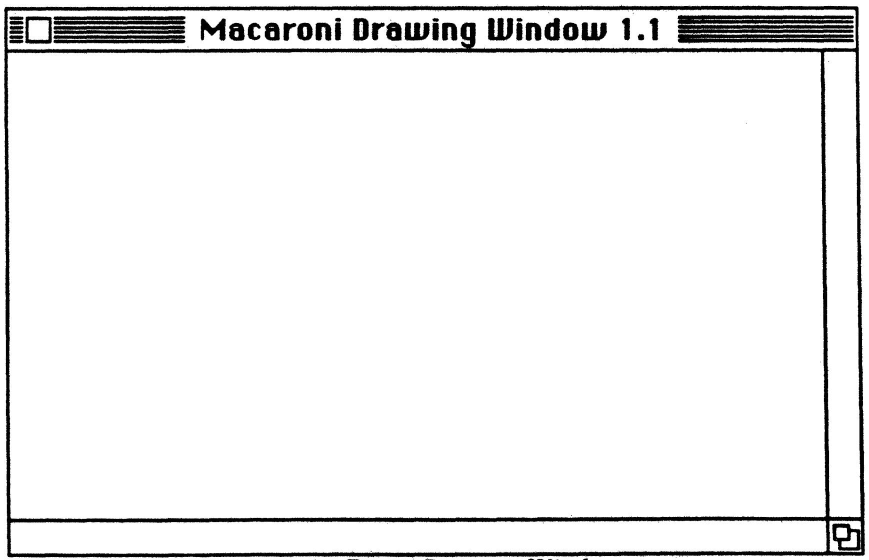
Resets the initial state of the pen in the current grafPort, as follows: pnSize to (1,1), pnMode to patCopy, pnPat to black. The pen location is not changed.

```
PenNormal();
```

The PenNormal Call

Click on the **EXECUTE** button and confirm the action.

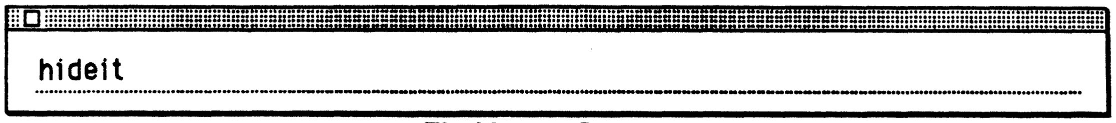
Macaroni will display an empty drawing window, which after resizing looks like this :



An Empty Drawing Window

If you have two monitors or a full or dual page display, pull the Drawing Window away from the browser.

If you have partial page display, pull down the message box from the HyperCard Go menu, type the following command and hit return to push the drawing window behind the browser:



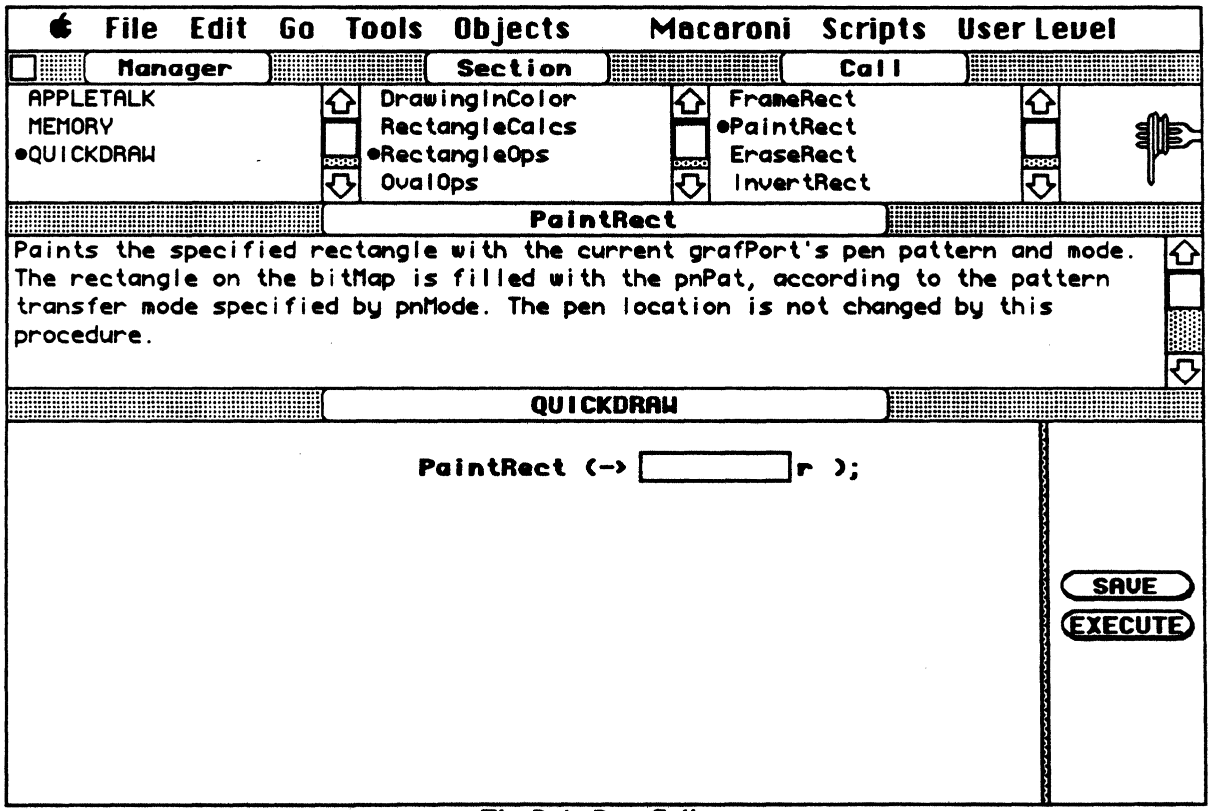
```
hideit
```

The Message Box

The *PaintRect* call paints a rectangle. To select the call do the following:

- Click on the word *Quickdraw* in the Manager Pane of the browser
  wait for Section Pane to update
- Click on the word *RectangleOps* in the Section Pane
  wait for the Call Pane to update
- Click on the word *PaintRect* in the Call Pane

Macaroni will display the following:



Paints the specified rectangle with the current grafPort's pen pattern and mode. The rectangle on the bitMap is filled with the pnPat, according to the pattern transfer mode specified by pnMode. The pen location is not changed by this procedure.

```
PaintRect (-> [          ]r );
```

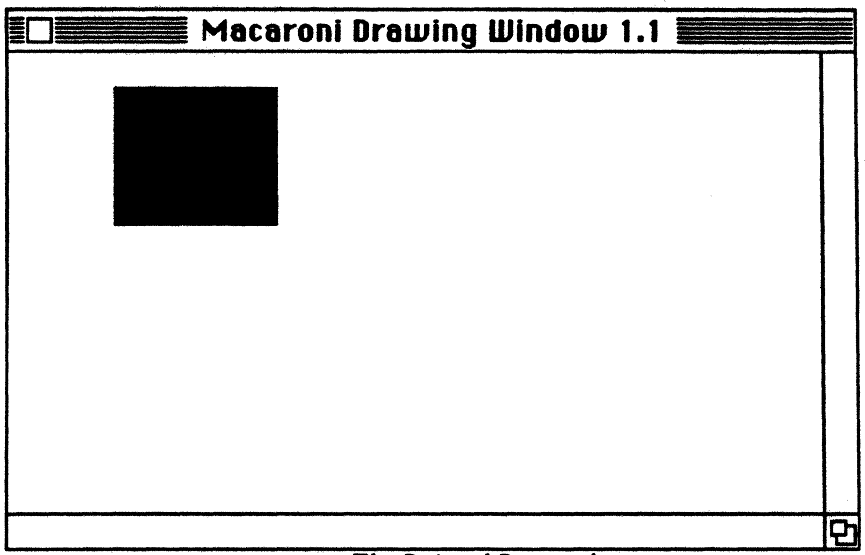
The PaintRect Call

Fill in the **RectPtr**:

-> 2873548 **RectPtr**

Click on the **EXECUTE** button and confirm the action.

Macaroni will paint the rectangle into the drawing window:



The Painted Rectangle

Congratulations!

You know how to use the Toolbox.

Disposing of a Pointer

Memory is released by calling the Memory Manager to dispose of the reference to the memory.

The physical memory addressed by the reference is returned to a free-list of memory blocks maintained by the Memory Manager and the reference becomes invalid.

Pointers are references to nonrelocatable memory. They are direct pointers to absolute addresses in some absolute address space. An absolute address points to a particular spot in physical memory or to a particular spot in an address space that is mapped into physical memory by a memory management unit.

DisposPtr is the Toolbox call that disposes of a pointer.

To display the *DisposPtr* call do the following:

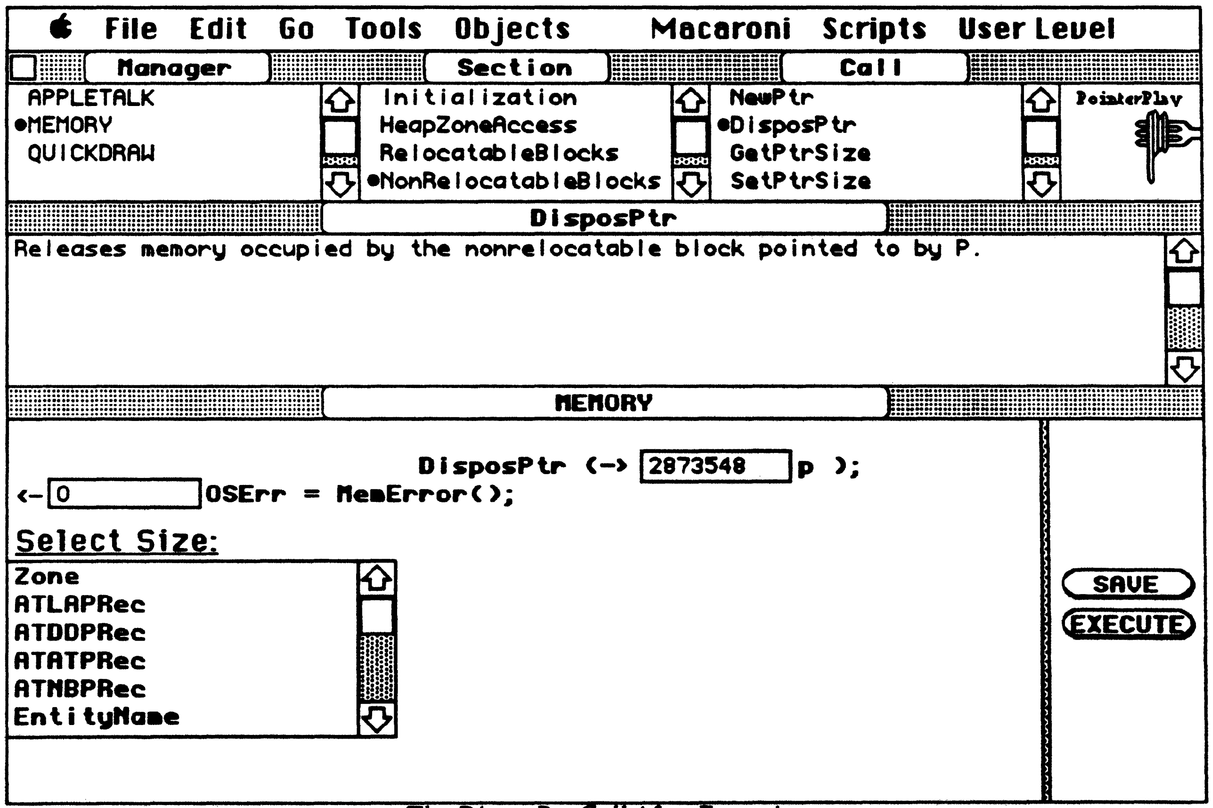
- Click on the word *Memory* in the Manager Pane of the browser
  wait for Section Pane to update
- Click on the word *NonRelocatableBlocks* in the Section Pane
  wait for the Call Pane to update
- Click on the word *DisposPtr* in the Call Pane

Macaroni will display the *DisposPtr* call.

To dispose of a pointer, and free its associated storage, enter the value of the pointer into the p field and click on the **EXECUTE** button. Confirm the **EXECUTE** in any dialogs that may appear.

On the top of the next page is a picture of the *DisposPtr* call after the **p** field has been filled in and the **EXECUTE** button has been pressed.

If you are working with Macaroni while you are reading this tutorial, make sure that you fill in the **p** field with the value of the pointer that you get from the *NewPtr* call.



Releases memory occupied by the nonrelocatable block pointed to by P.

```
DisposPtr (-> [2873548   ]p );
<-[0          ]OSErr = MemError();
```

The DisposPtr Call After Execution

Scripting *PointerPlay*

There are two techniques for entering calls into a Macaroni script. One technique is to save a call from the Toolbox Pane the other technique is to enter a call using the script editor.

Each of these techniques will be demonstrated using the following program which summarizes the preceding exercises:

```
on PointerPlay                               --declare the start of the script
  MacaroniBegin                              --prepare to enter the script
  put item one of NewPtr(8) into RectPtr     --allocate a Rect and get its pointer
  get SetRect(RectPtr,45,15,115,75)          --initialize the Rect
  get GetRect(RectPtr)                       --inspect the Rect
  get PenNormal()                            --set the pen to normal
  get PaintRect(RectPtr)                     --paint the rectangle
  get DisposPtr(RectPtr)                     --dispose of the Rect pointer
  MacaroniEnd                                --prepare to exit the script
end PointerPlay                              --declare the end of the script
```

Saving a Call

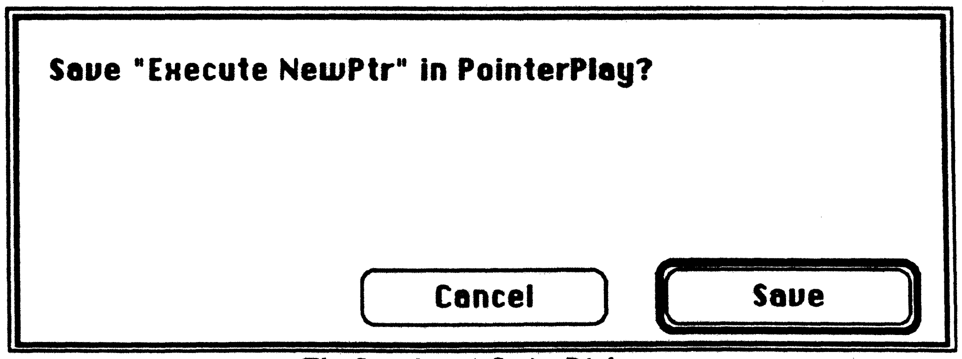
Any call shown in Macaroni can be saved into a script. This section of this manual, "Saving a Call", demonstrates how to save the *NewPtr* call into the *PointerPlay* script.

Display the *NewPtr* call by doing the following:

- Click on the word *Memory* in the Manager Pane of the browser
  wait for Section Pane to update
- Click on the word *NonRelocatableBlocks* in the Section Pane
  wait for the Call Pane to update
- Click on the word *NewPtr* in the Call Pane

Macaroni will display the *NewPtr* call.

Click on the SAVE button in the Control Pane that is adjacent to the call. Macaroni will display the following dialog:

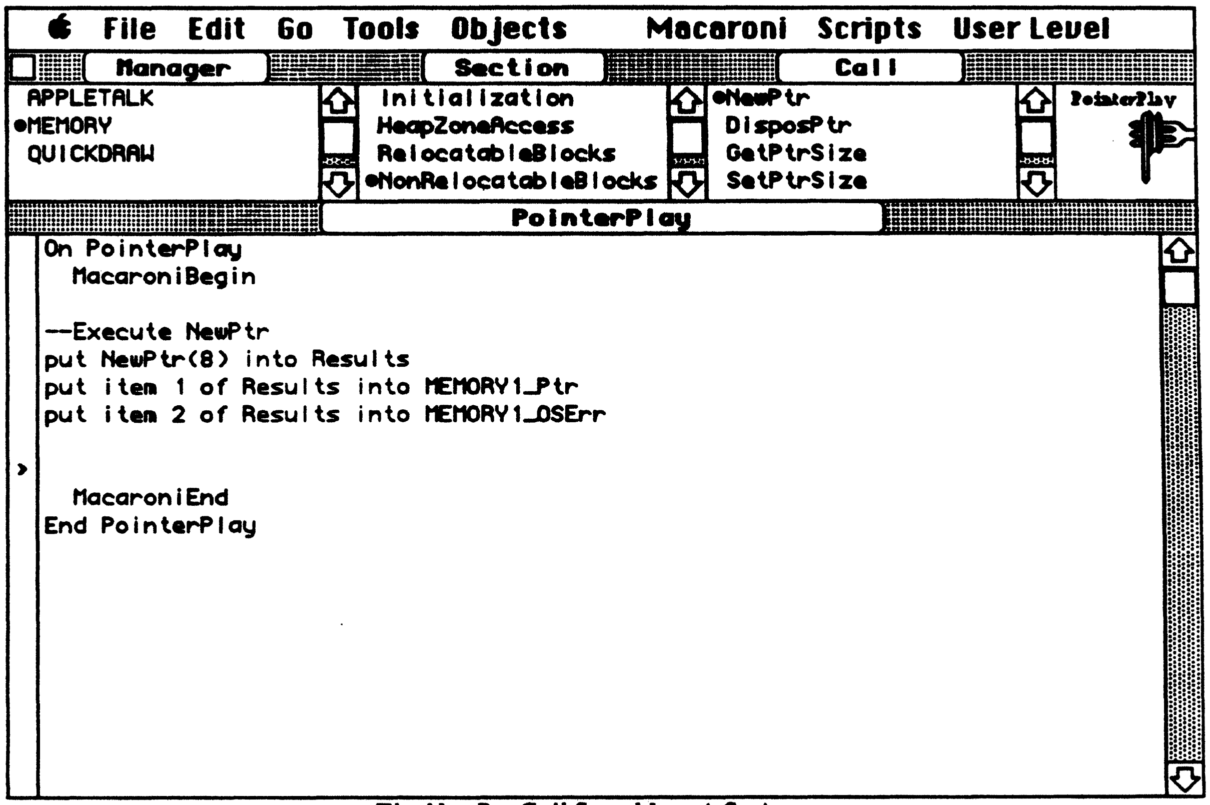

Save "Execute NewPtr" in PointerPlay?

The Save Into A Script Dialog

Confirm the dialog by clicking on the **SAVE** button and then display the *PointerPlay* script by clicking on the Macaroni button:

PointerPlay

Here is a picture of the script after the *NewPtr* call has been saved.



```
On PointerPlay
  MacaroniBegin

--Execute NewPtr
put NewPtr(8) into Results
put item 1 of Results into MEMORY1_Ptr
put item 2 of Results into MEMORY1_OSErr


  MacaroniEnd
End PointerPlay
```

The NewPtr Call Saved Into A Script

Look at the text of the call:

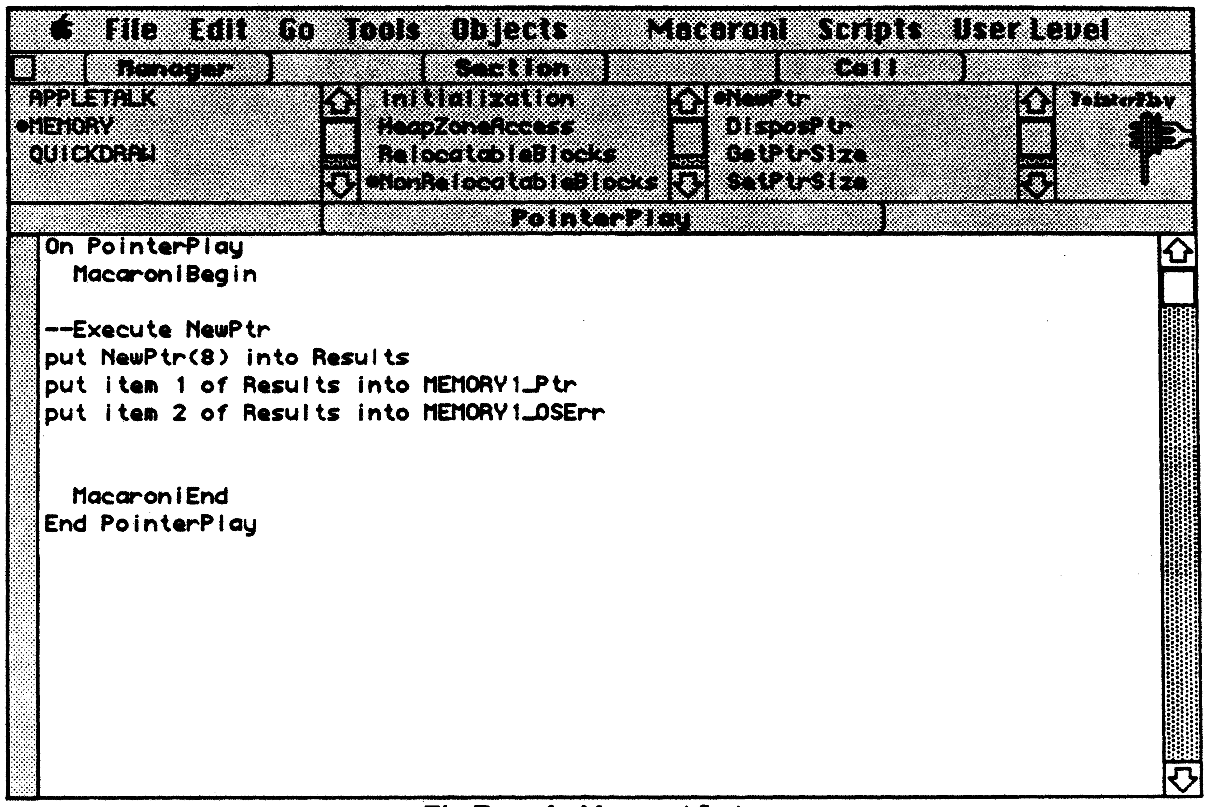
```
--Execute NewPtr
put NewPtr(8) into Results
put item 1 of Results into MEMORY1_Ptr
put item 2 of Results into MEMORY2_OSErr
```

Notice that Macaroni generates a name, MEMORY1_Ptr, that identifies the HyperTalk variable that is assigned the pointer to the rectangle.

The next section of this manual shows how to open the HyperTalk Script Editor and use it to enter the balance of the *PointerPlay* program into the *PointerPlay* script.

Editing the Script

To open up a script for editing, click on the text of the script, i.e. click anywhere in the region that is highlighted in the following picture:



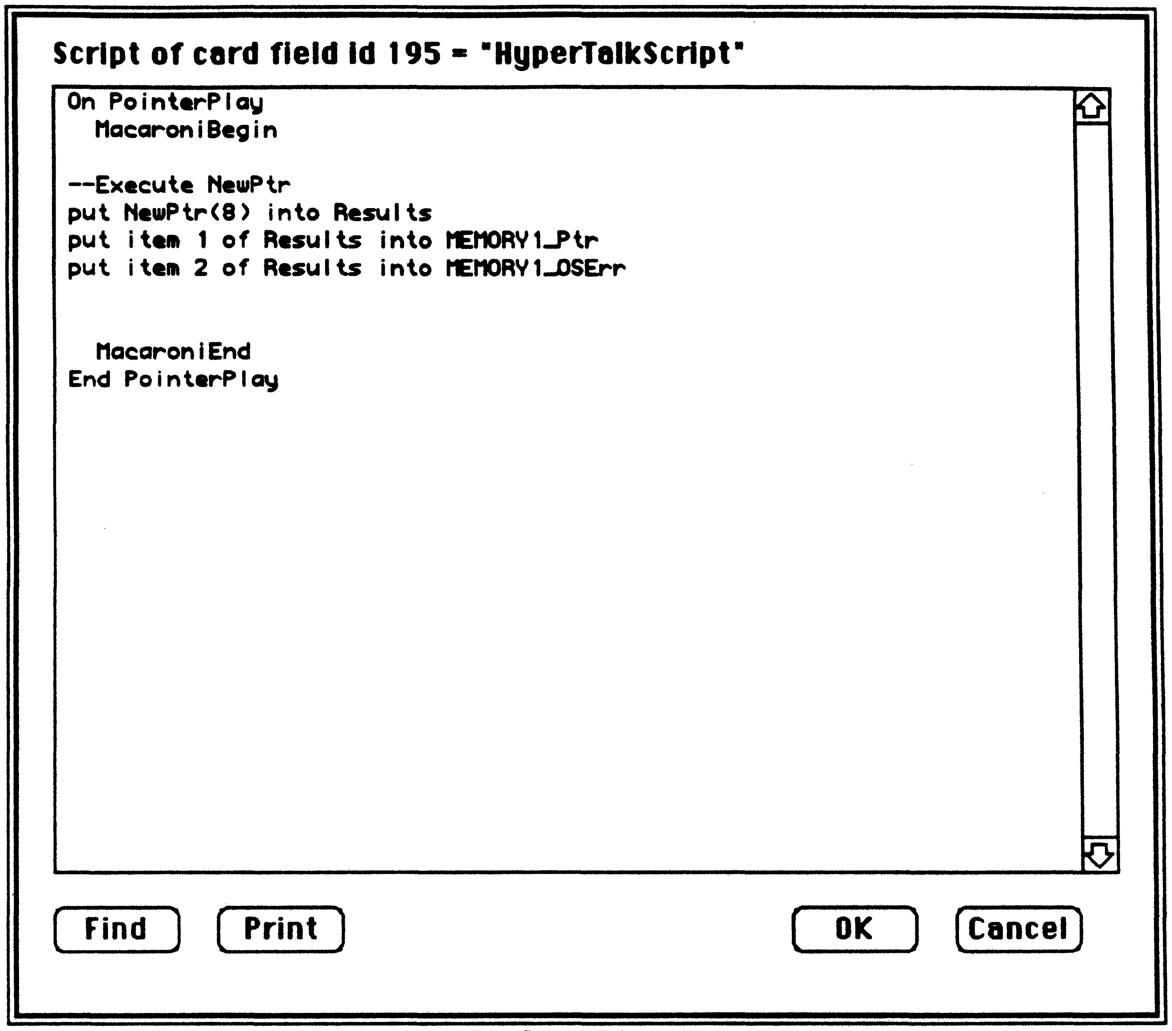
```
On PointerPlay
  MacaroniBegin

--Execute NewPtr
put NewPtr(8) into Results
put item 1 of Results into MEMORY1_Ptr
put item 2 of Results into MEMORY1_OSErr


  MacaroniEnd
End PointerPlay
```

The Text of a Macaroni Script

Macaroni opens the script up for editing and displays the script within the HyperCard Script Editor:



```
On PointerPlay
  MacaroniBegin

--Execute NewPtr
put NewPtr(8) into Results
put item 1 of Results into MEMORY1_Ptr
put item 2 of Results into MEMORY1_OSErr


  MacaroniEnd
End PointerPlay
```

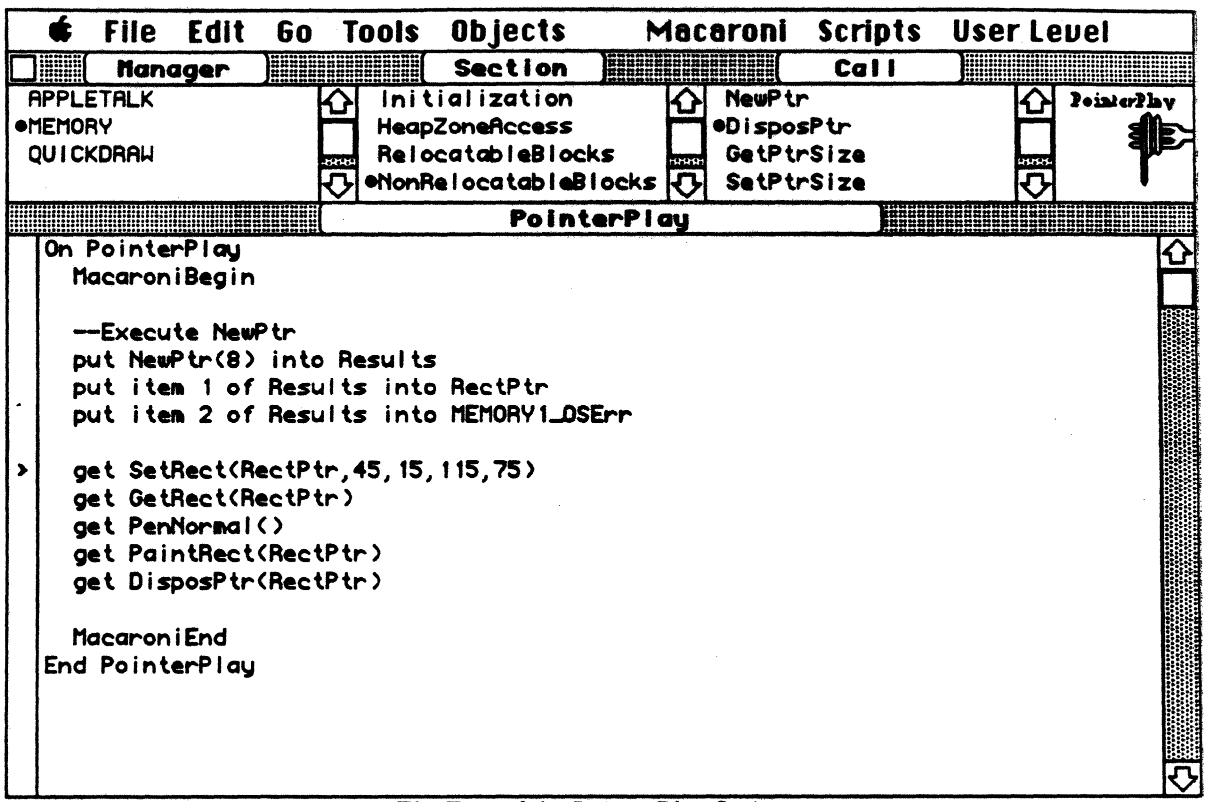
The Script Editor

Click in the script editor, change the name MEMORY1_Ptr to RectPtr and enter the following code between the *MacaroniBegin* and *MacaroniEnd* handlers:

```
get SetRect(RectPtr,45,15,115,75)
get GetRect(RectPtr)
get PenNormal()
get PaintRect(RectPtr)
get DisposPtr(RectPtr)
```

Click on the **OK** button to close the Script Editor and save the changes.

Macaroni will display the following:



```
On PointerPlay
  MacaroniBegin

  --Execute NewPtr
  put NewPtr(8) into Results
  put item 1 of Results into RectPtr
  put item 2 of Results into MEMORY1_OSErr

  get SetRect(RectPtr,45,15,115,75)
  get GetRect(RectPtr)
  get PenNormal()
  get PaintRect(RectPtr)
  get DisposPtr(RectPtr)

  MacaroniEnd
End PointerPlay
```

The Text of the PointerPlay Script

This is the end of the *PointerPlay* tutorial. You can either run *PointerPlay* or have fun exploring the Toolbox on your own.

The End

-- Staple Here --

Software Development Training Group

Steve Johnson

27 - ST

----------------------- Fold Here -----------------------

Macaroni Evaluation

Please identify the specific section or part of Macaroni as you complete the evaluation.

1. What's your overall rating of Macaroni on a scale of 1 to 5? ______
   (1 = Poor, 3 = Average, 5 = Excellent)

2. How was the technical level of Macaroni?
____too technical ____not enough ____just right ____other:_______

3. Was the self-paced content covered in sufficient depth? ____Yes ____No
If not, where was it not covered adequately?____
____
____

4. Were the stated objectives met by Macaroni? Why or why not?
____
____

5. Were the instructions and navigation through the tool adequate? Why or why not?
____
____

6. Were the related materials adequate? Why or why not?
____
____

7. Were there any parts of Macaroni you felt were poorly done?
____
____

8. What Macintosh Managers would you like to see added to Macaroni? (In priority order)
____
____

9. Please make any other comments you think will help us improve Macaroni (be as specific as possible):____
____
____
____

◆ Note: ***Please mail the completed evaluation form to the address indicated on back***

Read Me First

Macaroni: Macintosh Toolbox Training

What is Macaroni?

- Macaroni is an interactive stackware training tool for helping people learn the Macintosh Toolbox. Macaroni visualizes, in a simple and direct manner, the most popular calls from Apple's *Inside Macintosh*. Macaroni currently supports the AppleTalk, Memory and QuickDraw Macintosh Managers.

Who is supposed to use Macaroni?

- Employees in Apple R&D who have little or no Macintosh Toolbox specific experience.

What do I learn with Macaroni?

A student will be able to:

- Define the functionality of the Macaroni interface, which includes the Browser Pane, Toolbox Pane, the Control Pane, the Menu Bar and the Help System.
- Install, launch and quit Macaroni on your Macintosh system.
- Execute basic Macintosh Toolbox calls with Macaroni.
- Write, execute and edit a Macaroni recipe or script (a collection of Macintosh Toolbox commands).

How do I get started?

1) Open the User Manual called *Presenting Macaroni* available at the Engineering Support Library Open Document Shelves to page 1 to get an overview of Macaroni. Turn to page 2 to set up your Macintosh to run Macaroni.
Note: ***Your hard disk must have Hypercard 1.2.2 and at least 1.5 megabytes available storage to operate Macaroni.***

2) Read the Ingredients section, page 4, to learn the functionality and human interface of Macaroni.

3) **Copy the Macaroni 1.0ß2.2 folder onto your hard disk from the Engineering Support Fileserver.** Ask the ESL for the specific location. **Double-click Macaroni 1.0ß2.2 folder. Double-click Macaroni.**

4) Read the Techniques and Basic Recipes sections, pages 22 and 35 respectively, and follow along with the Macaroni software.

Where can I make comments or get more information?

- To provide comments, please fill out the enclosed evaluation or to get more information, please contact Steve Johnson at x4-6684.

◆ Note: ***This release of Macaroni is ßeta, so your comments are encouraged. Please fill out the evaluation form located in back of the documentation.***