PsyScope:

An Interactive Graphical System for Designing and Controlling Experiments in the Psychology Laboratory Using Macintosh Computers

Jonathan Cohen
Brian MacWhinney
Matthew Flatt
Jefferson Provost

Department of Psychology
Carnegie Mellon University

Running Head: PsyScope

Published in final form in Behavior Methods, Research, Instruments, and Computers, 25, 257-271 in 1993

Correspondence concerning this paper should be addressed via electronic mail to either Jonathan Cohen (jc5e+@andrew.cmu.edu) or Brian MacWhinney (brian+@andrew.cmu.edu), or via U.S. mail to either author at the Department of Psychology, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213.

Abstract

PsyScope is an integrated environment for designing and running psychology experiments on Macintosh computers. The primary goal of PsyScope is to give psychology students and trained researchers, alike, a tool that allows them to design experiments without the need for programming. PsyScope relies on the interactive, graphic enviornment provided by Macintosh computers to accomplish this goal. The standard components of a psychology experiment — groups, blocks, trials and factors — are all represented graphically, and experiments are constructed by working with these elements in interactive windows and dialogs. In this paper, we describe the overall organization of the program, provide an example of how a simple experiment can be constructed within its graphic environment, and discuss some of its technical features (such as its underlying scripting language, timing characteristics, etc.). PsyScope is available for non-comercial purposes free of charge and unsupported to the general research community. Information about how to obtain the program and its documentation is provided.

Introduction

The laboratory microcomputer has become a crucial tool for conducting experimental psychology. In recent years, a number of software programs have been developed on the Macintosh which rely on its user-friendly graphic interface to assist in the implementation of psychology experiments on microcomputers. However, for the most part these have been limited in range and/or power. For example, all of them lack a programming, or scripting language, and few of them have the capability to add external devices for more accurate timing of input and stimuli. There are powerful systems that run on other platforms. For example, Walter Schneider at the University of Pittsburgh has developed the most sophisticated of these — a system called the Microcomputer Experimental Laboratory (MEL; Schneider, 1988). However this runs only on IBM compatible PC's, and lacks the advantages of an easy to use, graphic interface. Vaughan (1992) has articulated the call for the development of an easy-to-use, intuitive environment in which psychologists — neophytes and experts alike — can develop experiments without the need for sophisticated programming skills.

We have developed a program, called PsyScope, which is intended to address many of these limitations. PsyScope has been developed at Carnegie Mellon University by Jonathan Cohen, Brian MacWhinney, Matthew Flatt, and Jefferson Provost. The primary goal of PsyScope is to eliminate the need for programming skills on the part of users, and to provide them with a set of tools that corresponds directly to the conceptual building blocks of experimental design: groups, blocks, trials, events (stimuli and responses) and factors. The program runs on Macintosh computers, and provides a graphic interface that allows the user to design an experiment, starting with the scientific question in mind, rather than being distracted by having to program it. This allows students to focus on

Figure 1

understanding the principles of Experimental Psychology, rather than the mechanics of experiment construction and computer programming. It also allows experienced researchers to implement experiments intuitively, using tools that correspond to the constructs to which they are accustomed.

Although PsyScope currently has many strengths, it would be misleading to suggest that, in its current form, it is a complete solution to all problems in experimental psychology. The goals of this paper are to provide an introduction to PsyScope — including the philosophy underlying its design the overall organization of the program — and to summarize some of its strengths and weaknesses.

The Philosophy Behind PsyScope

The basic philosophy underlying PsyScope is that improvements in the methodology of science inevitably lead to empirical and theoretical advances. In the case of experimental psychology, the introduction of microcomputer technology has been both a boon and a burden. It has been a boon in that it has allowed us far greater precision in experimental control and measurement. But it has become a burden in that it has forced many a budding psychologist to spend more time on programming than on experimental psychology. We believe that psychologists should be spending their time thinking and doing psychology rather than learning how to become computer programmers. Even those students and researchers who are accomplished programmers need to be able to move quickly between the conceptualization of a new experiment and its implementation.

Students or researchers who are not expert programmers often stumble on this hurdle. In this respect, non-programmer Psychologists are much like Chemists without bunsen burners or Geologists without rock hammers or compasses. The goal of PsyScope is to eliminate the need for programming skills on the part of users, and to provide them with a set of tools that correspond directly to the conceptual building blocks of experimental

Figure 2

design: groups, blocks, trials, events, and factors.   The program runs on Macintosh computers, and provides a graphic interface that allows the user to design an experiment, starting with the scientific question in mind, construct a full factorial elaboration of this experiment, and then pursue the articulation of the experiment by designing the exact stimulus types and the sequence of events for each cell in the factorial design.

The Overall Organization of PsyScope

While the goal of PsyScope is to eliminate the need for programming, we recognize that this goal may not be realistically — or at least not immediately — attainable. There is always a tension between the ease of use of structured, graphic environments and the power of a general purpose programming language. In order to make PsyScope as flexible and as generally applicable as possible, and to address the needs of more sophisticated users as well as non-programmers, we have built PsyScope around a general purpose scripting language. This language, called PsyScript, recognizes all of the experimental design structures that are supported in the graphic environment: groups, blocks, trials, events, and factors. In fact, when an experiment is constructed using graphical tools, PsyScope actually writes, and stores it as a script.   Scripts are text files, and can be edited directly in PsyScope, or using any standard text editor. Furthermore, the graphic and scripting environments are fully interactive. That is, any structure created in the graphic environment is immediately added to the script, and anything entered in the script that has a graphic representation appears immediately in graphical form. This interactivity has two valuable uses: 1) beginning scripters can use this as a learning tool, by observing the effects that adding or modifying graphic elements has on the script; 2) experienced scripters can take advantage of the convenience of the graphic environment to construct the basic outline of an experiment, eliminating repetitive typing, and then move to the script for the "detail work" needed to customize the experiment. PsyScope is also interactive in the sense that individual trials can be run as they are

Figure 3

constructed, singly or in blocks, so that the user can get immediate feedback about how the trial will appear when the experiment is run. In addition to the graphic environment and scripting language, there are a number of consoles and utilities that permit the editting of files (including log and data files) and that provide control over the running of the experiment.

In the following section, we provide an example of how to build an experiment using PsyScope's graphical tools. In later sections, we describe the scripting language and some of the utilities and technical features of PsyScope.

A Brief Example of Building an Experiment in PsyScope's Graphic Environment

When building a new experiment in the graphic environment, the user begins by using a palette of tools and objects in the Design window to link together icons that represent the components of the experiment.

Imagine that we want to construct a simple reaction time experiment to study the effects of distance from the fixation point and size of the target word on latency to word naming. We will call this the Acuity Experiment, and it will have two factors, Position and Size.

We begin by opening up the Design Window and linking a factor table object to the experiment (see Figure 1).

Figure 4

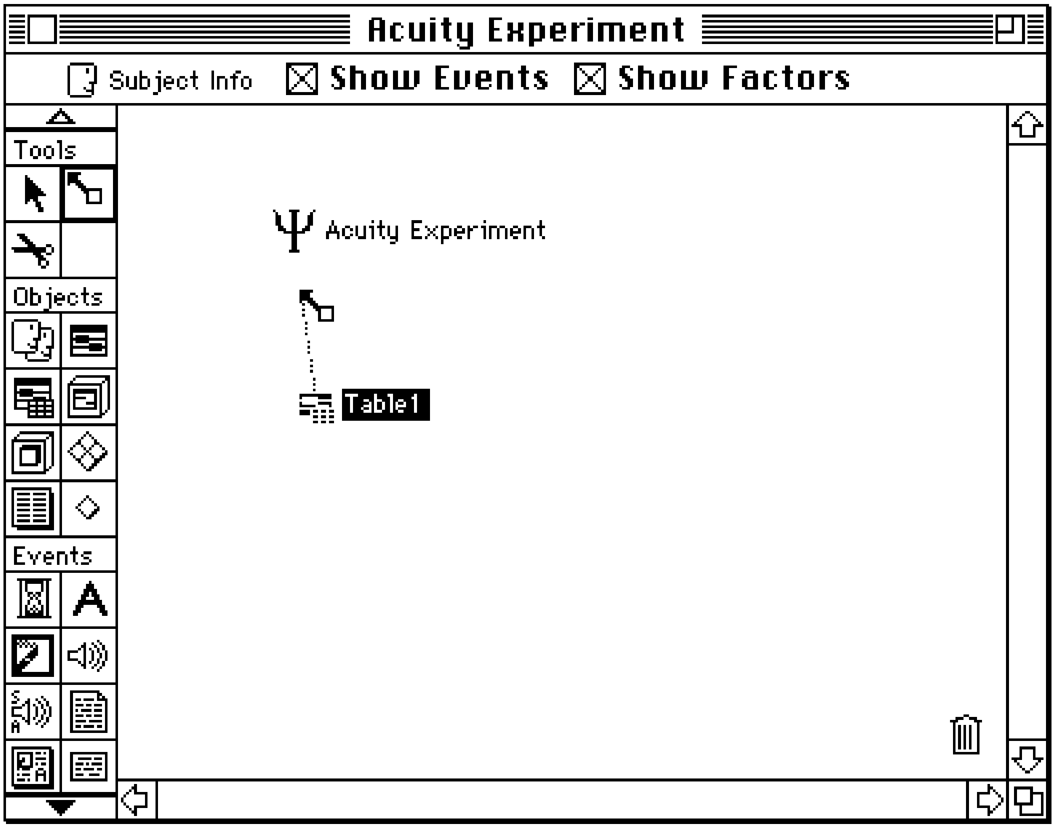
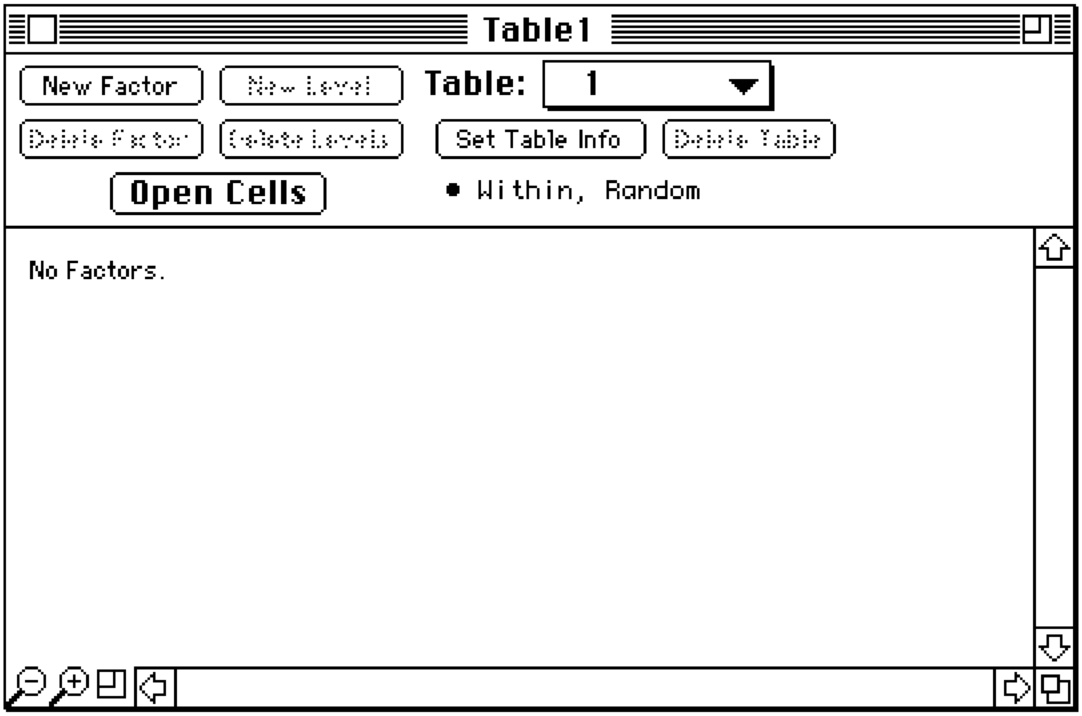


Figure 1. The Design Window. Experiments are constructed in the Design Window by linking graphical objects to the experiment. In this example, a factor table object is being linked to the experiment object.

Double-clicking on any of the icons in the Design Window opens up dialog windows that allow further elaboration of the design. For example, double-clicking on the Factor Table icon opens the factor table for this experiment (see Figure 2).

Figure 5



● Within, Random

No Factors.

Figure 2. The Factor Table window before any factors have been created.

Of course, it doesn't yet have any factors or levels. These have to be added. Our goal in this experiment is to construct a 2-factor experiment to measure the effects of distance (from fixation) and word size on naming latency. Our first factor is the position of the word, which will have five levels. To create this factor, we click on the New Factor button and then name it "Position." The factor appears in the table, and the New Level button is activated so that levels can be defined for the factor. Figure 3 shows how the factor table appears once the five levels for the Position factor have been created, and the Size factor and its 3 levels have been added.

Figure 6

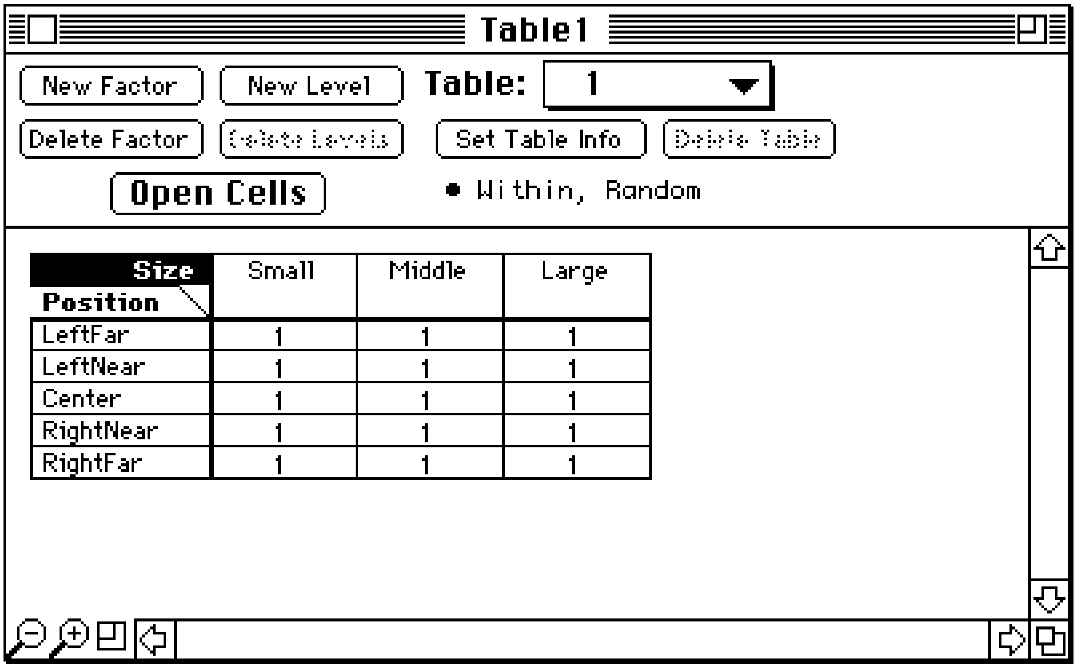


Table: 1

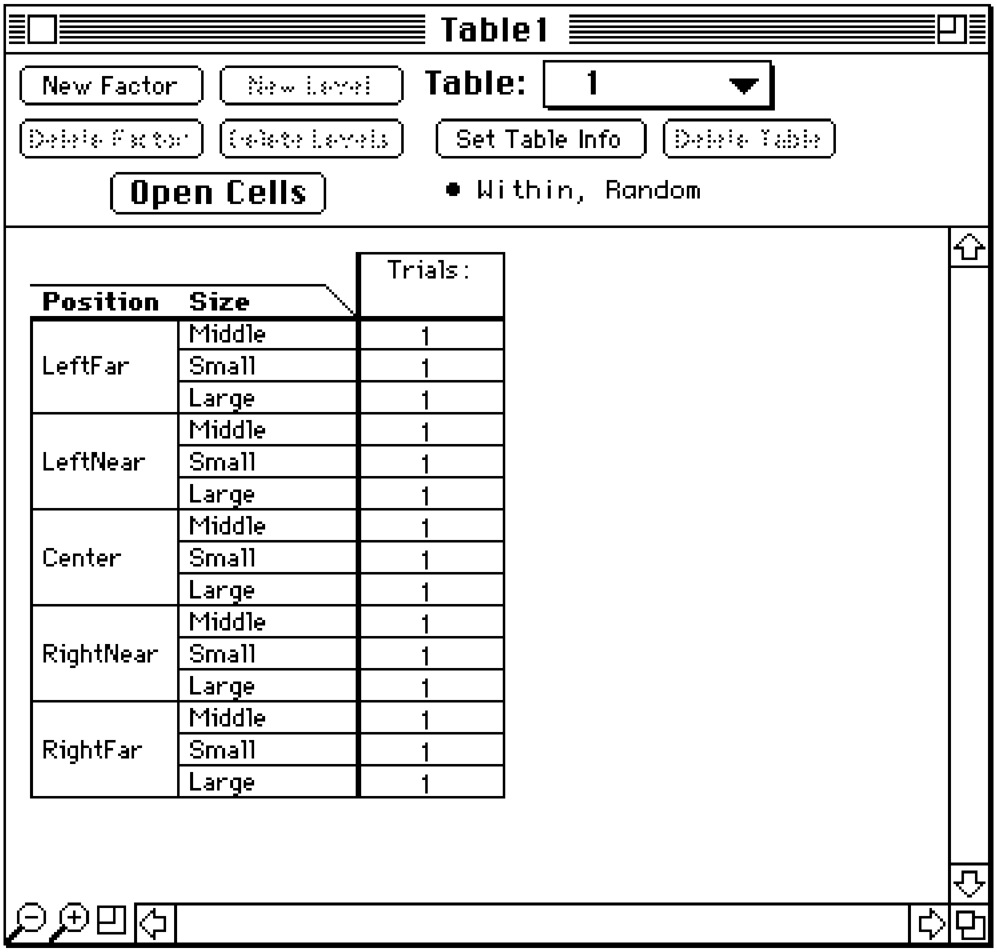
● Within, Random

Size / Position	Small	Middle	Large
LeftFar	1	1	1
LeftNear	1	1	1
Center	1	1	1
RightNear	1	1	1
RightFar	1	1	1

Figure 3. The factor table window once the Position and Size factors and their levels have been created. The numbers indicate the weight of each cell in the table, which is how many times that cell will be chosen on each pass through the table.

Factors and levels in a table can be re-arranged easily, by clicking and dragging. This allows the user to customize the display, and/or to change the order in which trials are run.

Figure 7



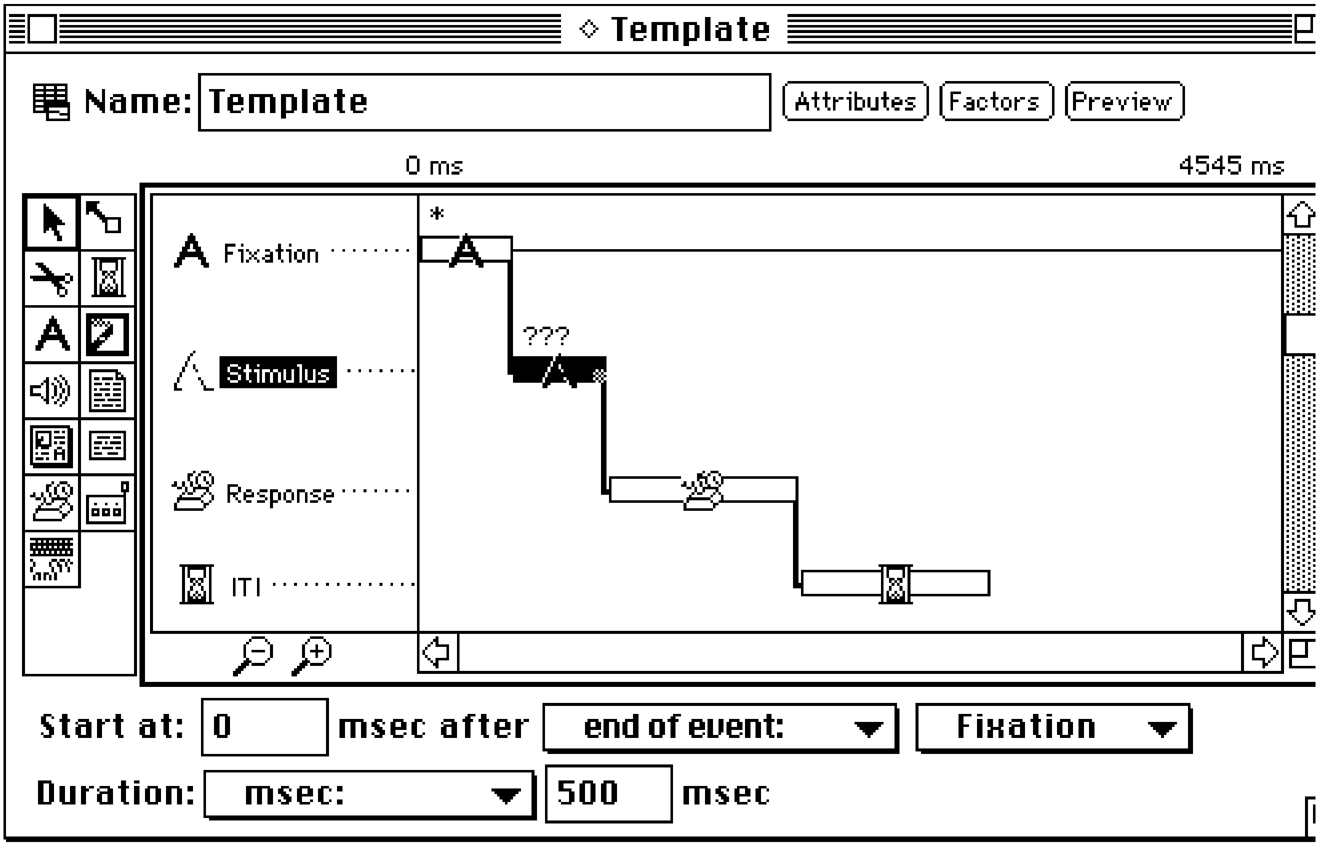
Position	Size	Trials:
LeftFar	Middle	1
	Small	1
	Large	1
LeftNear	Middle	1
	Small	1
	Large	1
Center	Middle	1
	Small	1
	Large	1
RightNear	Middle	1
	Small	1
	Large	1
RightFar	Middle	1
	Small	1
	Large	1

Figure 4. The factor table after the Size factor has been dragged to be a row, and the Small level of the Position factor has been dragged below the Middle level.

Once the factor structure has been established, the next step is to establish the structure of the trials that will instantiate each of the cells in the design. We begin by establishing the overall shape of the trials for all of the cells in the design. To do this, we can drag the mouse to select all of the cells, or simply type ⌘-A . Clicking on the Open Cells button, or double-clicking on the selected cells then opens the Template window. This window contains a palette of tools and objects for creating and linking the events that will make up a prototypical trial. An event is simply a period of time during which something

Figure 8

occurs. The events that make up a trial in the Acuity Experiment are the Fixation event (during which a fixation point is presented), the Stimulus event (which occurs at the end of the Fixation event), the Response event (during which the subject will respond), and the ITI event (the intertrial interval). These are created and represented graphically along a time line in the Template window (see Figure 5).



Name: Template

Start at: 0 msec after end of event: Fixation

Duration: msec: 500 msec

Figure 5. The Template window after the events for a prototypical trial have been created. The Stimulus Event is selected. Notice the menus at the bottom of the window for setting the start time and duration for the event.

Figure 9

Each type of event is represented in the palette with its own icon. The set of events standardly available in PsyScope is listed in Table 1. In addition, new types of events can be created and added to PsyScope as code resources. However, this requires programming expertise (see below).

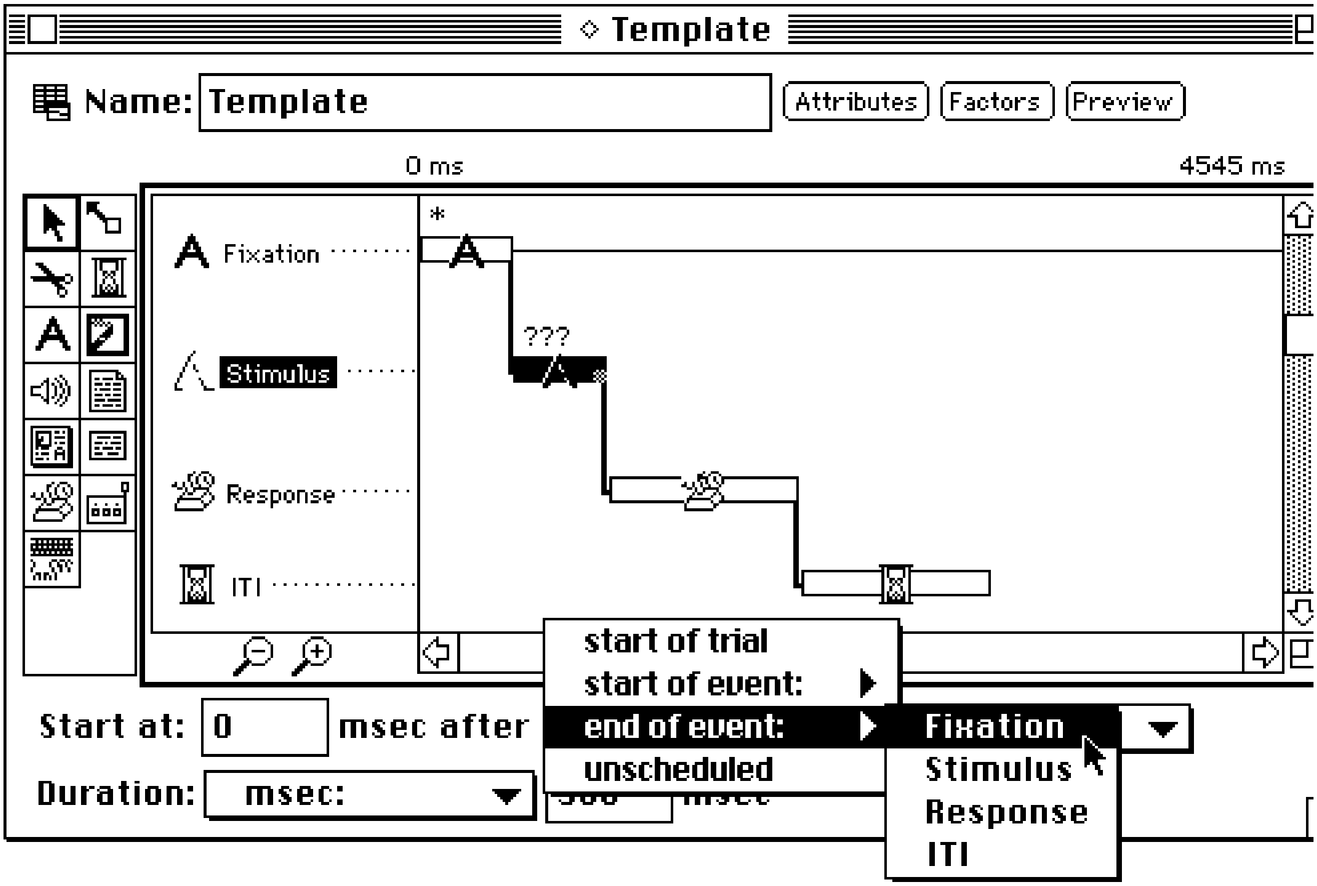
Table 1. Standard PsyScope Event Types.

Icon	*Event Type*	*Description*
⌛	Time	Waits for a timed period.
A	Text	Presents free unformatted text.
▤	Paragraph	Presents formatted text.
🗎	Document	Presents the full contents of a text file.
🖼	PICT	Presents a graphic contained in a file in PICT format.
📋	Pasteboard	Presents any combination of text and/or PICTs simultaneously. These can be positioned relative to one another.
🔊	Sound	Presents a sound contained in a file in SoundEdit format. The entire file or just a labeled sound can be presented.
📟	Bbox Output	Turns one of the three LED lights on the CMU button box on or off, or changes the status of one of the 8 TTL output lines.

Figure 10

	Input	Records a single mouse click, a single key press, or a single response from the CMU button box (a button press or a signal from the voice activated relay).
	Continuous Keyboard Input	Records a string typed at the keyboard.

Each event is associated with a set of attributes, which are used to specify exactly how that event will occur. The start time and duration of the event can be set directly from the Template window, by selecting the event and using the pop-up menus at the bottom of the window (see Figure 6).



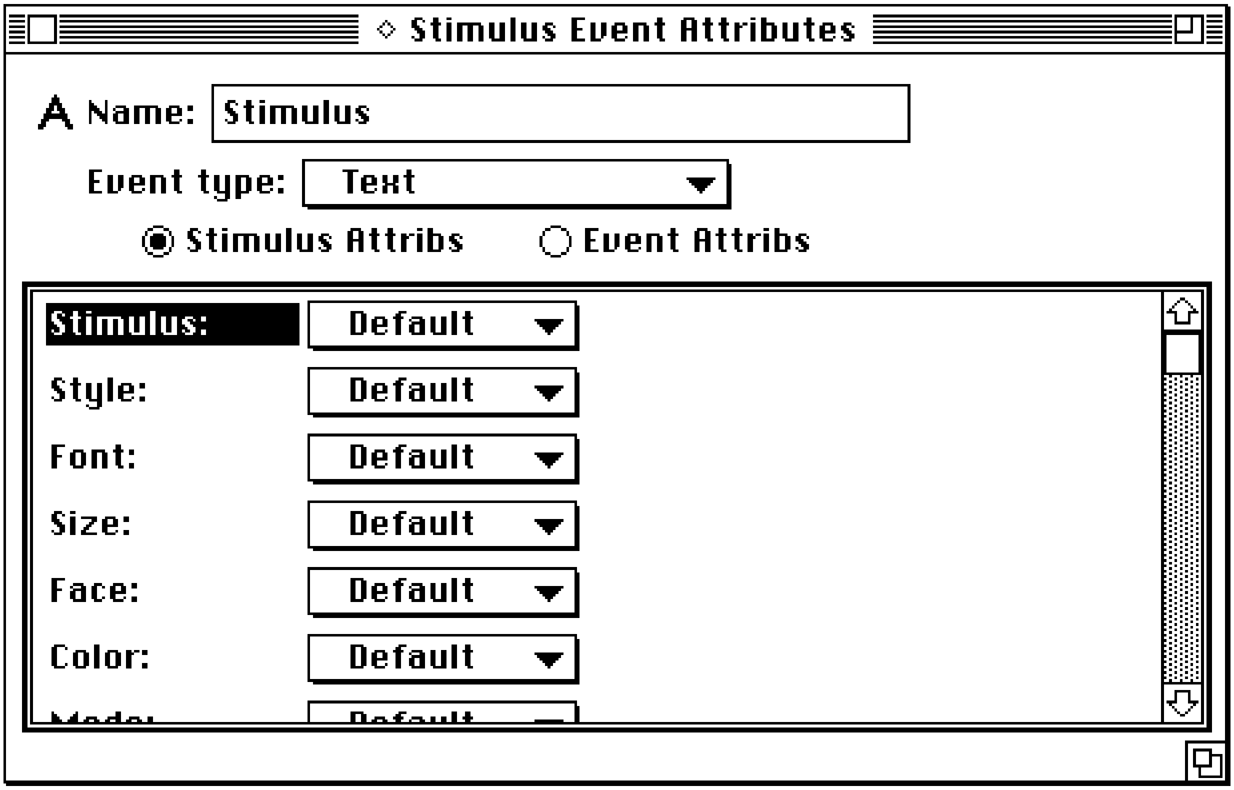
Start at: 0 msec after

Duration: msec:

Figure 6. The pop-up menus for setting the start time of the Stimulus Event.

Figure 11

The other attributes for the event are set in the Attributes dialog, which is opened by double-clicking on the event in the Template window. Each attribute has a pop-up menu that is used to assign its value (see Figure 7).



◇ Stimulus Event Attributes

Name: Stimulus

Event type: Text

● Stimulus Attribs ○ Event Attribs

Stimulus: Default

Style: Default

Font: Default

Size: Default

Face: Default

Color: Default

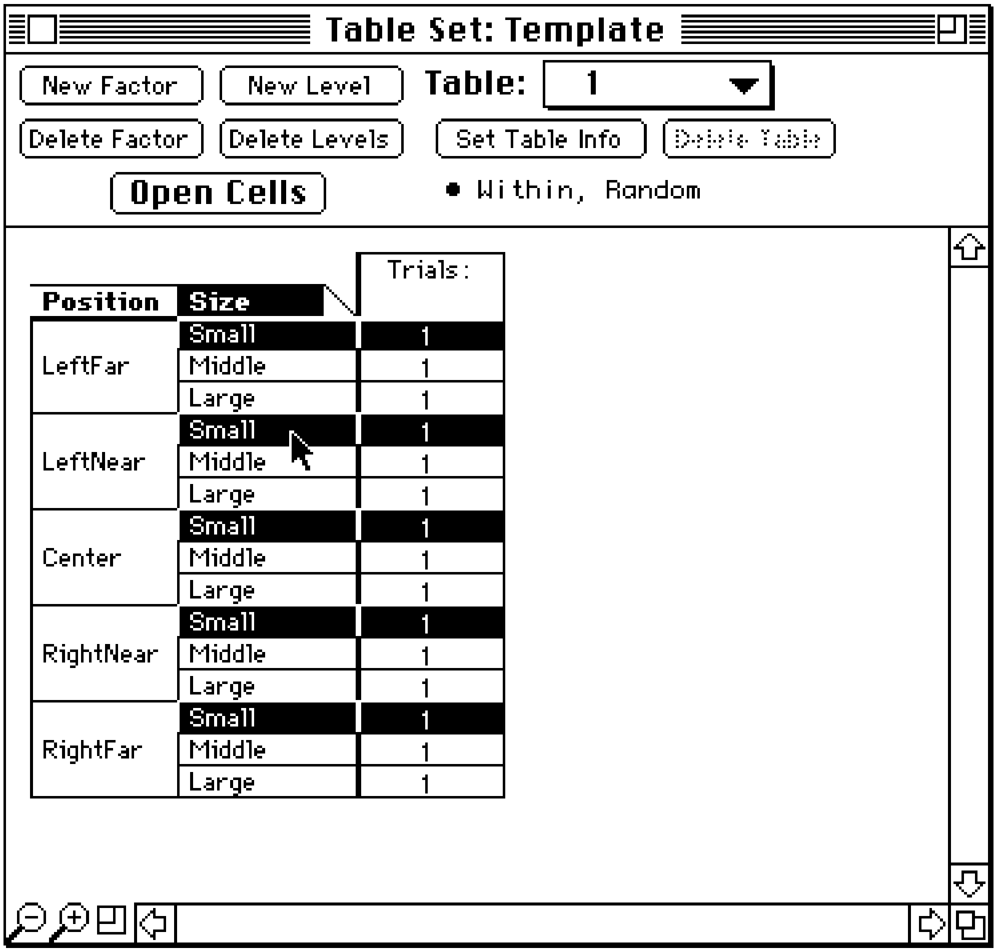
Mode: Default

Figure 7. The Attributes dialog for the Stimulus event, showing the “Vary by” submenu.

In the Acuity Experiment, we want the value of the Size and Position attributes for the Stimulus event to be determined by the factorial design of the experiment. To do this, we return to the factor table. Beginning with the Size factor, we select the cells for the first level by clicking on "Small" (see Figure 8). The Template window and Attribute dialogs are all linked to the Factor Table window, so that when a subset of cells is selected in the factor table, any changes made in the Template window or Attributes dialogs will apply

Figure 12

only to those cells of the table. That is, the characteristics of the trial can now be customized for each cell, to suit the factorial design of the experiment. Any feature of the trial can be changed, and new ones can be added, for any single cell or set of cells in the factor table.

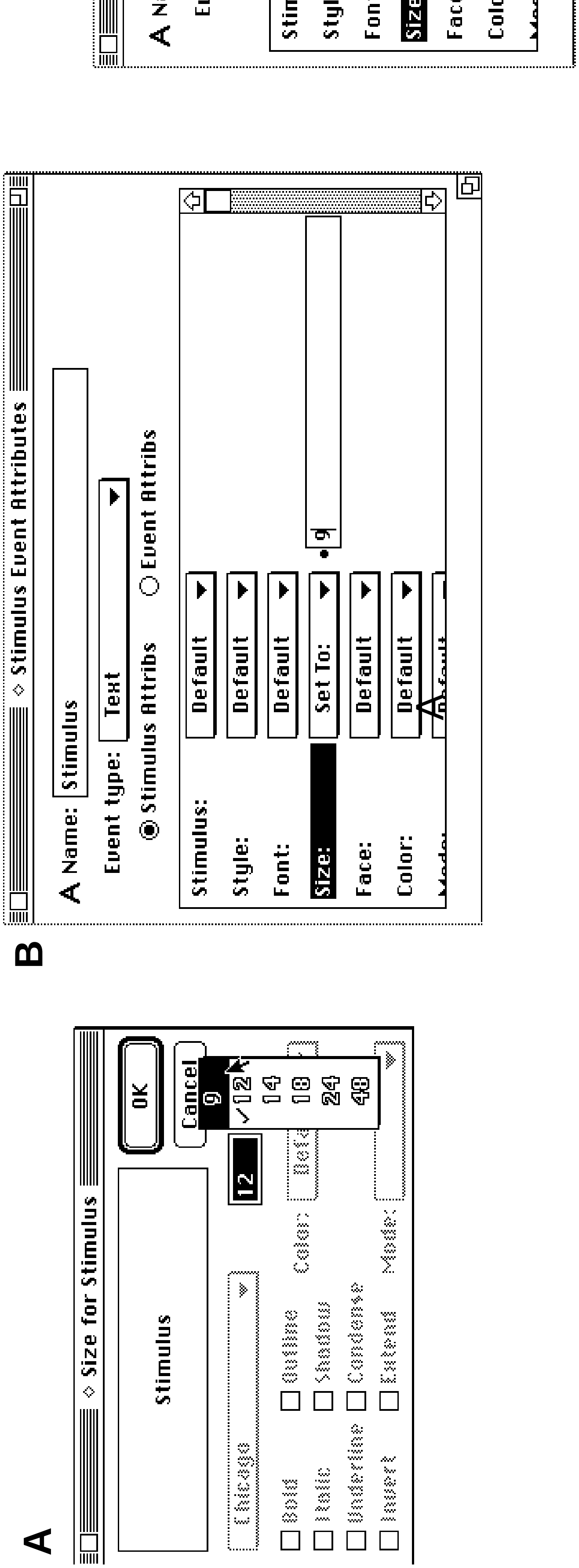


Position	Size	Trials:
LeftFar	Small	1
	Middle	1
	Large	1
LeftNear	Small	1
	Middle	1
	Large	1
Center	Small	1
	Middle	1
	Large	1
RightNear	Small	1
	Middle	1
	Large	1
RightFar	Small	1
	Middle	1
	Large	1

Figure 8. Clicking on the heading for a particular level of a factor selects all of the cells for that level.

Having selected the cells for the Small level of the Size factor, we now go to the Attributes dialog for the Stimulus event and set the Size attribute to some small size, say 9 point. To do this, we choose Set To from pop-up menu for the Size attribute. This opens a dialog that allows us to set the size interactively (see Figure 9a); alternatively, we can use a short cut to enter the value as text (see Figure 9b).

Figure 13



◇ Size for Stimulus

Stimulus

Chicago

□ Bold □ Outline
□ Italic □ Shadow
□ Underline □ Condense
□ Invert □ Extend

Color: Default

Mode:

12

◇ Stimulus Event Attributes

Name: Stimulus

Event type: Text

● Stimulus Attribs ○ Event Attribs

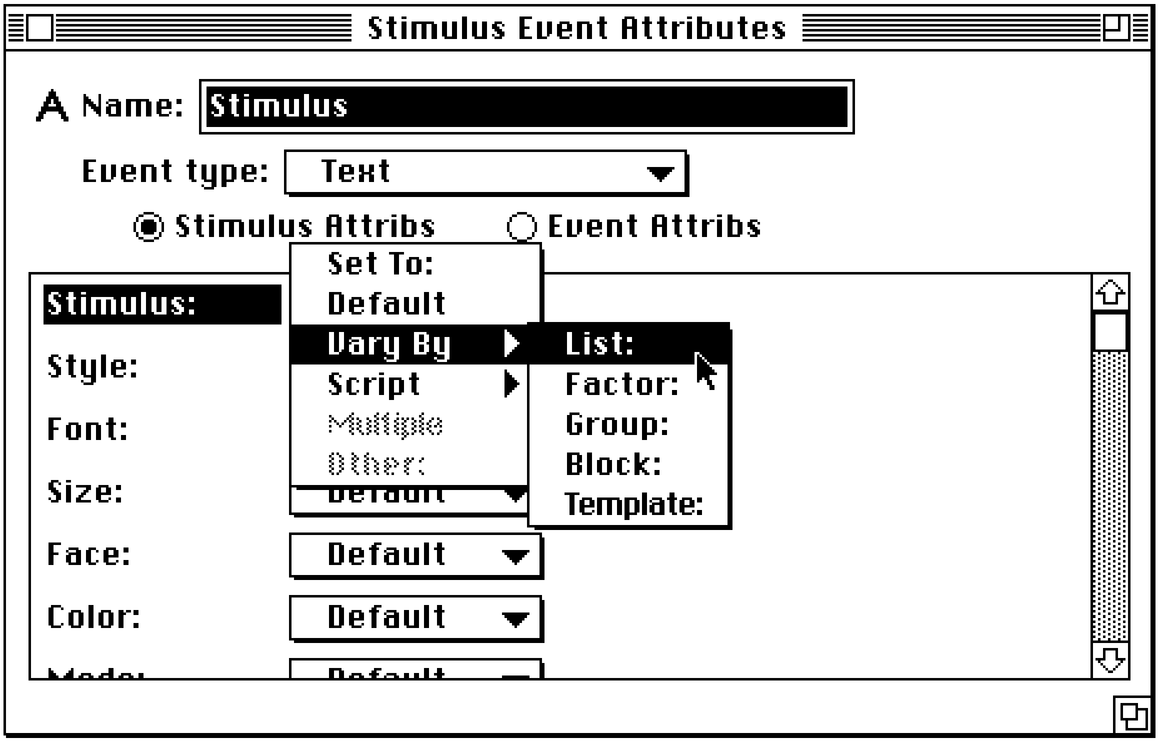
Stimulus: Default
Style: Default
Font: Default
Size: Set To: 9
Face: Default
Color: Default
Mode: Default

Figure 9: Two methods for setting attributes

Figure 14

We repeat this series of steps to assign the values for the Middle and Large levels of the Size factor, and then do the same for the Position factor, in this case setting the value of the Position attribute.

Now that the basic structure of the experiment has been set up, and the values have been assigned for the levels of each factor, we need to specify the actual stimuli that will be presented. In this experiment, we will simply choose words randomly from a list. To do this, we select all of the cells in the factor table, and then go to the Attribute dialog for the Stimulus event. There, we choose "Vary by" from the Stimulus attribute pop-up menu and "List" from the "Vary by" submenu (see Figure 10).



Stimulus Event Attributes

Name: Stimulus

Event type: Text

● Stimulus Attribs ○ Event Attribs

Stimulus:

Style:

Font:

Size: Default

Face: Default

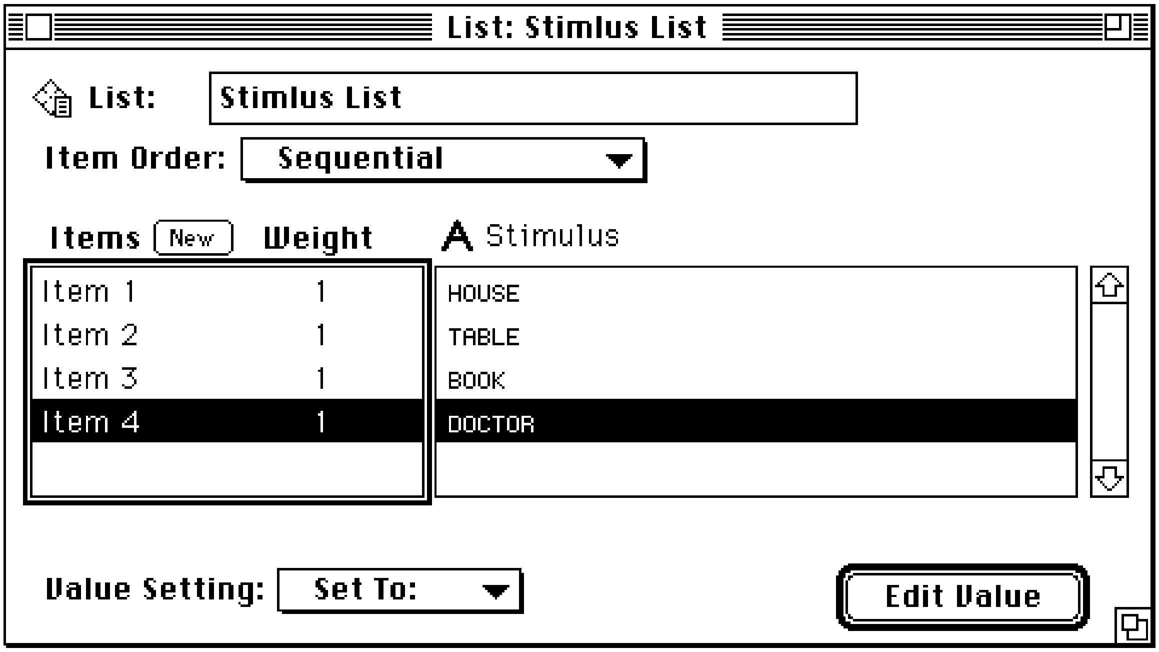
Color: Default

Mode: Default

Figure 10. Choosing "Vary by" and "List" from the Stimulus attribute menu causes the stimulus for each trial to be chosen from a list.

Since no list exists yet, this will open the List dialog, which allows us to create a new one (see Figure 11).

Figure 15



List: Stimlus List

List: Stimlus List

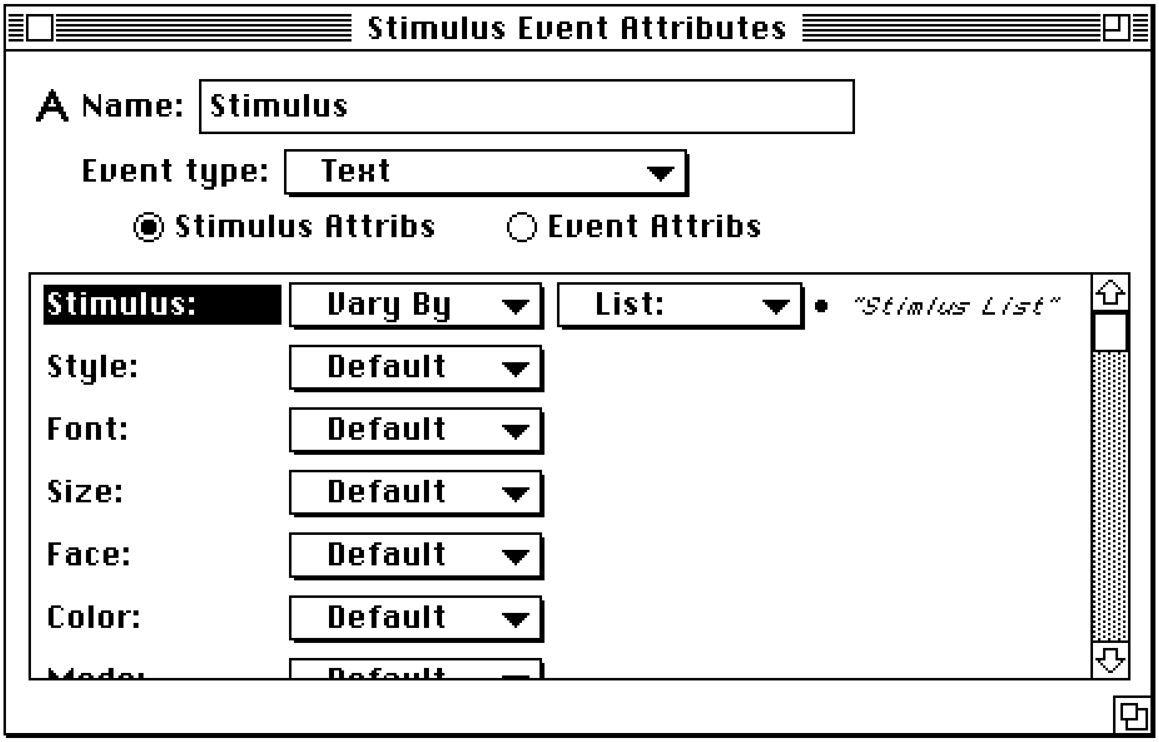
Item Order: Sequential

Items	Weight	Stimulus
Item 1	1	HOUSE
Item 2	1	TABLE
Item 3	1	BOOK
Item 4	1	DOCTOR

Value Setting: Set To:

Figure 11. The List dialog, once a few items have been created.

When we are done, and close the List dialog, the Attribute dialog will reflect that the stimulus for the Stimulus event will be chosen, trial to trial, from the list (see Figure 12).



Stimulus Event Attributes

Name: Stimulus

Event type: Text

● Stimulus Attribs ○ Event Attribs

Stimulus:	Vary By	List:	• "Stimlus List"
Style:	Default		
Font:	Default		
Size:	Default		
Face:	Default		
Color:	Default		
Mode:	Default		

Figure 16

Figure 12. The Attribute dialog for the Stimulus event, indicating that the stimulus attribute will be varied by the Stimulus List.

After a few more dialogs to specify how responses will be recorded, to set up some instructions and a rest period, and to block the trials, we have completed the design of the experiment. Figure 13 shows how it will appear in the Design Window.

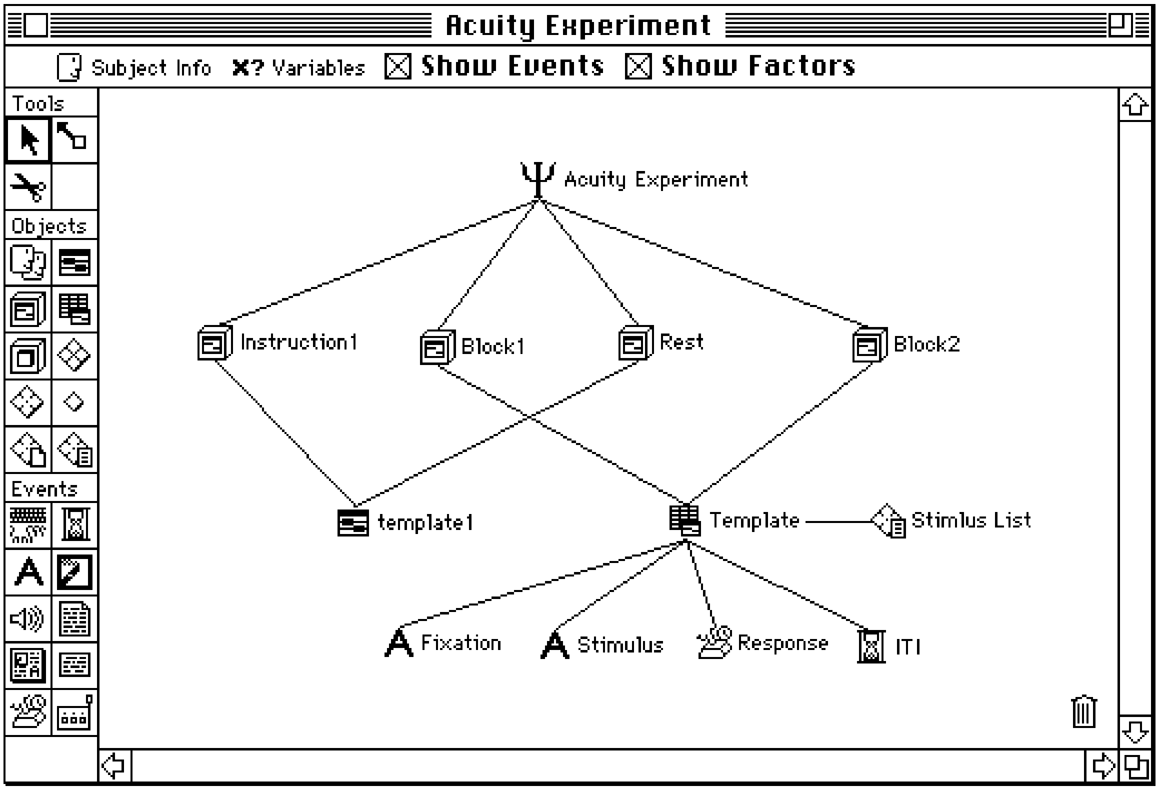
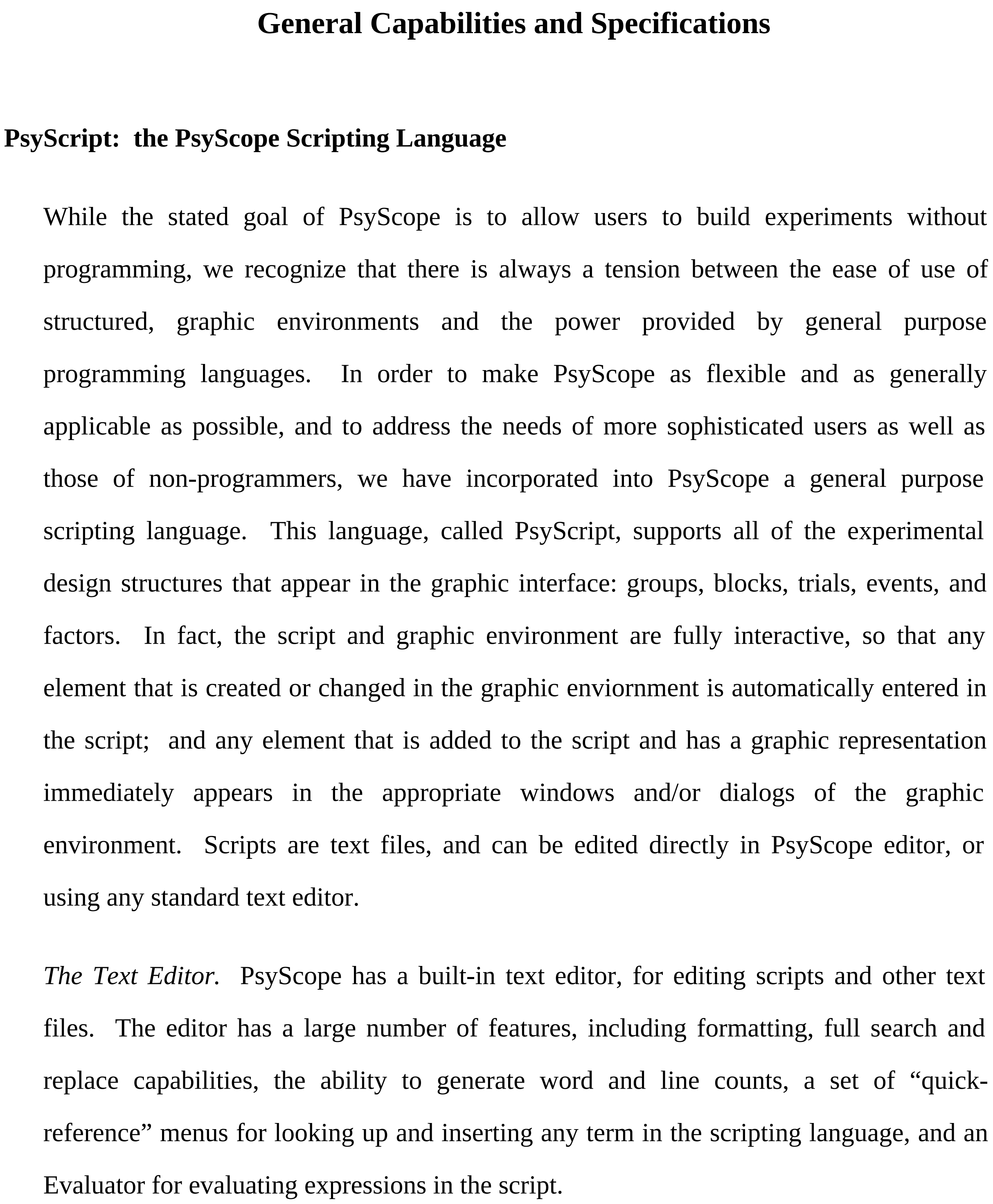


Figure 13. The Design window showing the complete experiment. Notice that the events and the list all appear as elements of the experiment.

We can now run the experiment, by choosing Run from the Run menu, clicking on the Run button in the Console window, or typing ⌘-R. The responses for each trial will be recorded in a standard tab-delimited text file. We can also preview trials at any point during the design process, by clicking on the Preview button in the Template window (to preview a single trial) or in the Block dialog (to preview a block of trials).

Figure 17



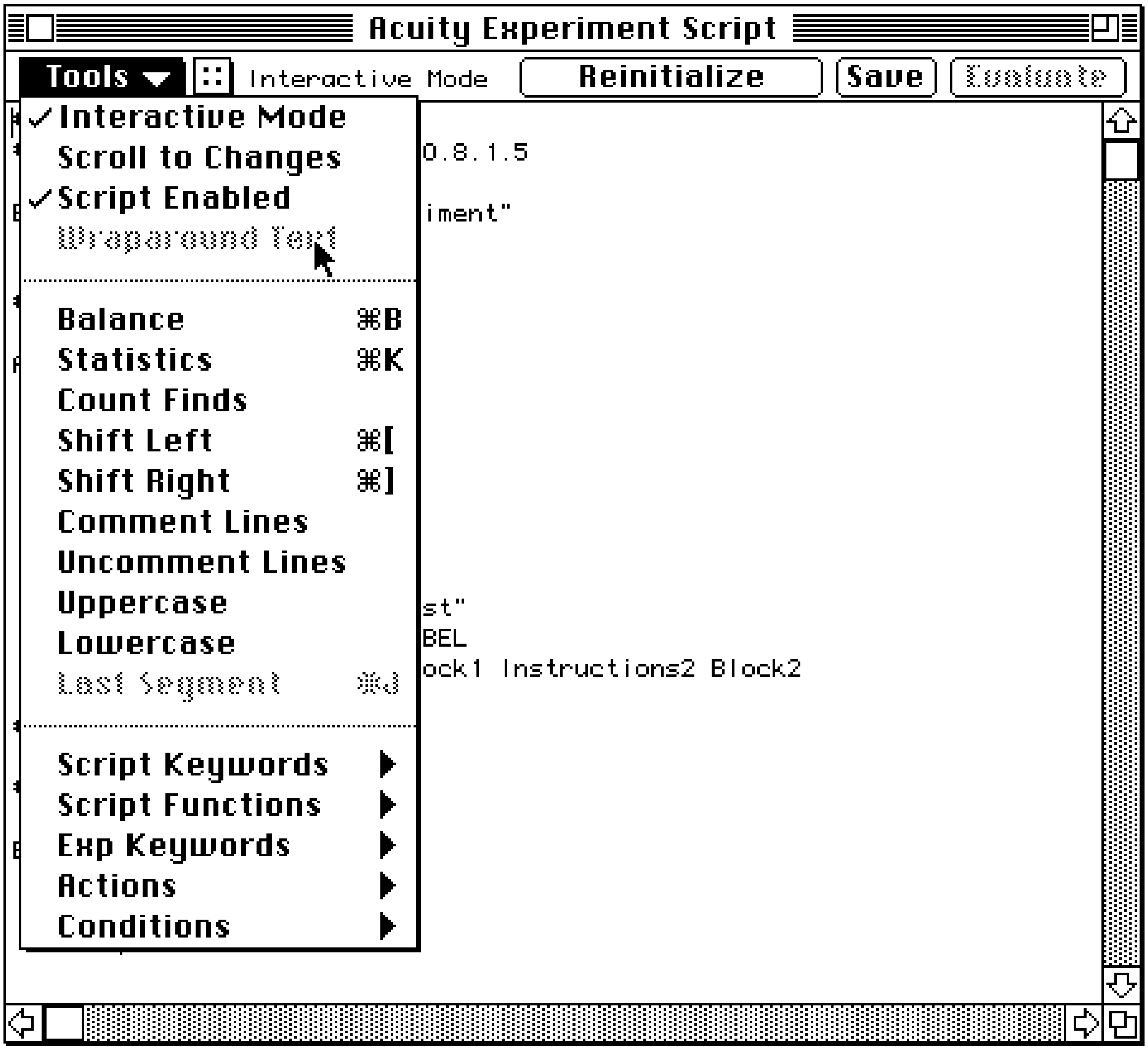
General Capabilities and Specifications

PsyScript: the PsyScope Scripting Language

While the stated goal of PsyScope is to allow users to build experiments without programming, we recognize that there is always a tension between the ease of use of structured, graphic environments and the power provided by general purpose programming languages. In order to make PsyScope as flexible and as generally applicable as possible, and to address the needs of more sophisticated users as well as those of non-programmers, we have incorporated into PsyScope a general purpose scripting language. This language, called PsyScript, supports all of the experimental design structures that appear in the graphic interface: groups, blocks, trials, events, and factors. In fact, the script and graphic environment are fully interactive, so that any element that is created or changed in the graphic enviornment is automatically entered in the script; and any element that is added to the script and has a graphic representation immediately appears in the appropriate windows and/or dialogs of the graphic environment. Scripts are text files, and can be edited directly in PsyScope editor, or using any standard text editor.

The Text Editor. PsyScope has a built-in text editor, for editing scripts and other text files. The editor has a large number of features, including formatting, full search and replace capabilities, the ability to generate word and line counts, a set of "quick-reference" menus for looking up and inserting any term in the scripting language, and an Evaluator for evaluating expressions in the script.

Figure 18



```
0.8.1.5
iment"
st"
BEL
ock1 Instructions2 Block2
```

Figure 14. The Editor window, showing the Tools menu.

Brief Description of PsyScript. PsyScript is best described as a declarative language. It is used to define the structure of the experiment in hierarchical form: An experiment is composed of groups, a group is composed of blocks, a block is composed of trials (templates), and trials are composed of events. Each of these, and all other components of PsyScope, are specified as "entries" in the script. Each entry has a title, and a set of attributes that define its characteristics. The entry for each component in the experiment hierarchy has one attribute which specifies the components that belong to it (i.e., are

Figure 19

below it in the hierarchy). Figure 15 shows fragments from two parts of the script for the Acuity Experiment.

Figure 20

```
Acuity Experiment::
      Format: Factor
      InputDevices: MOUSE KEY
      Timer: Macintosh
      Flags: NO_SAVE_SCREEN
      DataFile: "acuity.data"
      ScaleBlocks: 1
      Cycles: 1
      Instructions:"acuity.inst"
      DataFields: RESPONSE_LABEL
      Blocks: Instruction1 Block1 Instructions2 Block2

#> BlockDefinitions

Block1::
      Cycles: 10
      Templates: Template

Block2::
      templates: Template
      BlockDuration: 15

Instruction1::
      templates: Instruction Template

Instructions2::
      templates: Instruction Template
```

```
#> EventDefinitions

Fixation::
      EventType: Text
      Duration: 500
      StartRef:
      Stimulus: "*"

Stimulus::
      EventType: Text
      Duration: 500
      StartRef: "0 after end of Fixation"
```

Figure 21

Figure 15. Two script fragments from the Acuity Experiment script. The one on the left shows the entries for the experiment and two of the blocks in the experiment (corresponding to the graphical elements in the Design window — see Figure 13); the one on the right shows the entries for the Fixation and Stimulus events (which are displayed graphically in the Design and Template windows — see Figures 13 and 5, respectively).

There is a basic, implicit procedure in PsyScript which is the instantiation of experimental trials from the statements in the script. The compiler begins with the top level entry in the script (the experiment entry), and works its way down the hierarchy looking to see what components it is made up of (groups, blocks, templates, and events, etc.) until all of the events of each trial in the experiment are fully defined. Any undefined elements are assigned predefined default values (except the stimulus for each event, which must be defined). Variable expressions and function calls are permitted at any level, providing extensive flexibility in the definition of the experiment. In particular, PsyScript provides a powerful set of list operations (e.g., accessing lists, sublisting, mapping between lists, etc.) that assist in the development of highly sophisticated randomization designs.

Script Compilation and Run Modes

The script is compiled when the experiment (or a subset of trials) is run. This can be done in one of two modes: either one trial at a time (“trial-by-trial mode”), or all at once before any trials are run (“precompile mode”). When PsyScope compiles a trial, it determines and sequences all of the events for that trial, and assigns the values for all of their attributes. These are written in a memory structure, which is then read by the Trial Manager that executes the trial in real time (timing is discussed in the next section).

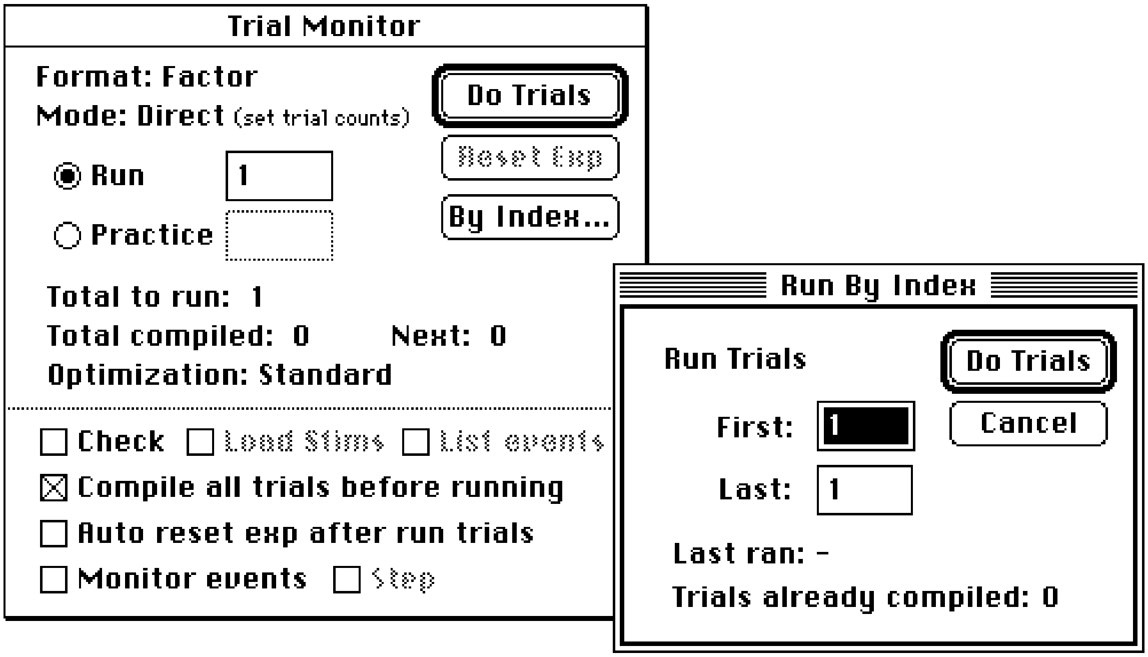
Figure 22

Trial-by-trial Mode. In this mode, PsyScope compiles a single trial, and then runs it before compiling the next. The primary advantage of this mode is that features of the experiment can change as it is proceeds — for example, the duration of a stimulus can change from one trial to the next, based on previous reaction times. The disadvantage of this mode is that the compile time may vary from trial to trial, introducing variability in the intertrial interval. This can be overcome in one of two ways: either by explicitly including an intertrial interval that is longer than the maximum amount of time it will take to compile a trial (this can be done by setting an experiment attribute), or by precompiling all of the trials before running the experiment.

Precompile Mode. In this mode, PsyScope reads the entire script and constructs all of the trials of the experiment before any is run. This mode optimizes performance at the time the experiment is run (eliminating any unspecified intertrial intervals), but this is at the expense of the flexibility offered by the trial-by-trial mode.

Previewing and Pre-checking Trials. In addition to the two general run modes just described, single trials or sets of trials can be compiled and run using the Trial Monitor console (see Figure 16), or by using the Preview buttons in the Template window and Block dialogs. These provide the user with immediate feedback about how trials will appear when the experiment is run. The Trial Monitor can also be used to monitor the events of a trial as it runs (for debugging scripts), and to pre-check the experiment (e.g., to see how long it takes to compile trials, to be sure that all stimulus files are correctly referenced, etc.).

Figure 23



Trial Monitor

Format: Factor
Mode: Direct (set trial counts)

● Run 1
○ Practice

Total to run: 1
Total compiled: 0 Next: 0
Optimization: Standard

□ Check □ Load Stims □ List events
☒ Compile all trials before running
□ Auto reset exp after run trials
□ Monitor events □ Step

Run By Index

Run Trials
First: 1
Last: 1
Last ran: -
Trials already compiled: 0

Figure 16. The Trial Monitor console, and the Run By Index dialog that can be opened from it.

Randomization

PsyScope supports a wide variety of randomization designs — including mixed vs. blocked, full factorial, nested factorial, and between subject (including Latin Squares) designs. There are also facilities for defining groups for between subject designs. All of the above are available within the graphic environment, often with a single click of the mouse. More complex designs can also be developed in the graphic environment, though this requires more experience with PsyScope. In addition, PsyScript can be used to construct designs and randomizations of arbitrary complexity.

Data output

PsyScope provides no facilities for statistical analysis. There are currently a wide variety of statistical packages available for the Macintosh. Rather than trying to duplicate these efforts within PsyScope, our goal has been to make available as much information about

Figure 24

the experiment as possible available for inclusion in the data file — in standard text format — and then allow the user to choose which of this information to actually record.

A data file consists of a list of data lines with a fixed number of tab-delimited columns. A data line is generated during a trial under conditions that are specified by the user. Typically, data lines are scheduled to occur in response to subject input (a key press or a mouse click, etc.). However, data lines can be generated at any other point during a trial (the beginning or end of an event, when a user-defined variable reaches some value, etc.). Each data line contains at least three columns, which record the number of the trial, the name of the event, and the condition that generated it. In addition, the user can choose to include columns for text labels, experimental condition name (generated from the factor table), the time of occurrence (relative to the beginningof the trial or some event) as well as number of other items.. A header can also be added to the data file, which can include information about the time and date of the run, the type of machine on which it took place, and subject identification information. Finally, PsyScope has a built-in facility for recording information about each session (such as subject and run information and error messages) in a separate log file. This is also stored as a text file, which can be changed at any time, and used by muliple experiments.

Timing

PsyScope provides a minimum of 16 ms accuracy (the temporal resolution of the Macintosh operating system) in the timing of events when run on a free-standing Macintosh. However, it has the capability to provide one millisecond accuracy when used with the CMU button box (see next section) or another external timing device. An extension must be written to use an external timing device (see below); once this is written, it can be added to PsyScope simply by dropping it in the PsyScope Extensions

Figure 25

folder. The timing of stimulus presentation and response recording are, of course, limited by the input and output devices that are used.

CMU Button Box

This is a device that has been developed at CMU for use in microcomputer controlled psychological experiments. It can be used with Macintosh or IBM-PC compatible computers. It connects to the Macintosh via the modem or printer port, and provides a millisecond timer, as well as three colored buttons with associated LED lights, a voice activated relay, 4 additional TTL input lines and 8 TTL output lines. The extension for this device is built into PsyScope, allowing PsyScope to recognize all of its input and output lines, as well as providing PsyScope with millisecond timing accuracy. This device will be fully described in a separate paper that is currently under preparation.

Input and Output

Input. PsyScope can record mouse clicks or movement, single or multiple key presses, and input from the CMU button box (described above). All input signals (including mouse and keyboard input) are recorded asynchronously, so that multiple simultaneous inputs can be recorded (within the temporal resolution provided by the hardware).

Output . PsyScope can present 1 to 24 bit visual stimuli to any standard display that is connected to a Macintosh, and to any or all of the displays in a multiple-display system. Stimuli presented on the screen are linked to the vertical retrace (VBL) signal by default, but this can be defeated by setting an experiment attribute. Formatted text as well as graphics stored in PICT format (either in a file or resource) can be displayed. PsyScope can play 8 bit sounds stored as SND resources or in SoundEdit format. The LED lights and output lines of the CMU button box can also be controlled. PsyScope uses standard Macintosh Toolbox calls to construct and present all stimuli to their corresponding output devices. Stimuli are presented asynchronously, permitting multiple different stimuli to be presented at the same time. The only exception to this is refresh-synchronized screen displays, which require PsyScope to wait for the VBL signal (0-16 ms). However, multiple, refresh-synchronized stimuli can be presented simultaneously, by using the PasteBoard event type. This composes the stimuli into a single bitmap before writing to the screen, waits for the VBL signal, and then presents the entire bitmap at once.

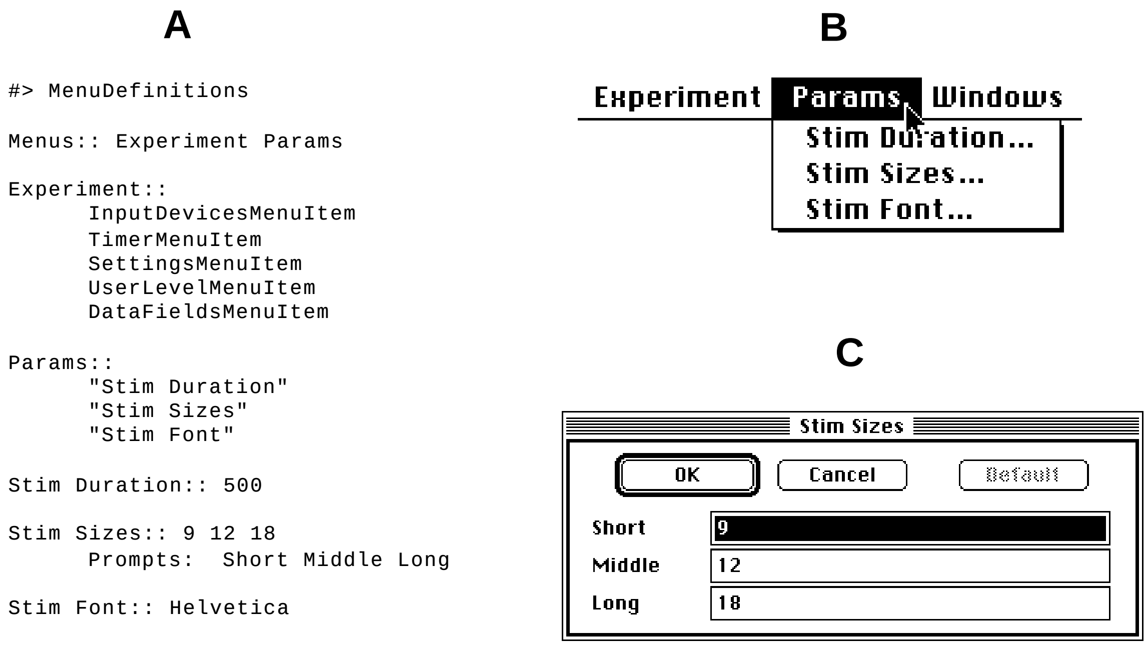
In addition to the standard input and output devices, PsyScope can be extended to work with other devices by writing externals, which are discussed in the next section.

Customizing PsyScope

There are two ways in which PsyScope can be extended: using the scripting language to customize the interface, and by writing externals.

Customizing the Interface. PsyScope provides built-in tools for extending the interface to include new interactive elements, such as menus and dialogs. These can be used to conveniently modify experimental parameters at the time the experiment is run (e.g., the duration of an interstimulus interval), or to record information that needs to be entered at the time the experiment is run (e.g., subject id, choice of conditions, etc.). These are defined in the same way that the components of an experiment are defined, by including new menus and dialogs in entries, and specifying their contents or format in attributes of the entry. Figure 17 shows a script fragment that adds a custom menu to the menu bar.

Figure 27



```
#> MenuDefinitions

Menus:: Experiment Params

Experiment::
      InputDevicesMenuItem
      TimerMenuItem
      SettingsMenuItem
      UserLevelMenuItem
      DataFieldsMenuItem

Params::
      "Stim Duration"
      "Stim Sizes"
      "Stim Font"

Stim Duration:: 500

Stim Sizes:: 9 12 18
      Prompts:  Short Middle Long

Stim Font:: Helvetica
```

Short 9
Middle 12
Long 18

Figure 17. (A) Script fragment showing the addition of Params as a custom menu, which has three items: Stim Duration, Stim Sizes, and Stim Font; (B) the Params menu; (C) the dialog opened by the choosing the “Stim Sizes” item from the Params menu.

Externals: the PsyScope Extensions folder. PsyScope also provides the ability — for C programmers — to write extensions to the program itself, called PSYXs. Conventions for writing PSYXs (similar to HyperCard XCMDs) are documented in the PsyScope manual. PSYXs can be written to control new input and/or output devices, to connect new timing devices, to carry out specialized functions that can be called from the scripting language, and to add new dialog or graphical elements to the interface. PSYXs can also be written to replace the compiler that reads the script and creates the memory strcture for the experiment, permitting the design of experiments in a language other than PsyScript. Once a PSYX is written, it is added to PsyScope simply by dropping it into the PsyScope Extensions folder. No recompiling of the program is needed: The extension will become active the next time PsyScope is started. PSYXs have recently been written to send and receive Apple Events (Apple's standard for interapplication communication on the Macintosh), and for the DigiDesign 16 bit sound board, and

Figure 28

extensions are currently under development for continuous microphone recording (straight to disk) and for an oscilloscope-based device that provides visual displays with 1 ms temporal resolution.

Help and Documentation

PsyScope has an integrated help system. Help can be called from the script or the graphic environment, and provides a search ability and index.

The PsyScope manual is still under development. It currently consists of five parts. Parts I through III and Part V are complete. These are a general introduction and tutorial for using the graphic environment (Part I), a reference manual for the graphic environment which includes a full description of every window and dialog (Part II), a reference manual for all of PsyScope's main menus and general utilities (Part III), and the PsyScript reference manual (Part V). Part IV (an introduction to scripting experiments) and Part VI (the reference manual for writing PSYXs) are in intermediate stages of completion. The manuals contain extensive figures showing the screens and graphic elements used in PsyScope. All documents are in Microsoft Word format.

There is also a growing library of commented example scripts, which implement a variety of standard psychology experiments (e.g., Stroop, the Sperling task, lexical priming, etc.). These are available to members of the PsyScope Development Consortium (see below).

System and Hardware Requirements

System. PsyScope requires at least 2MB of RAM and about the same amount of hardisk space. This is sufficient to construct and store a moderately complex experimental design. More complex designs, or experiments involving many stimuli may require more memory and disk space.

Figure 29

CPUs. PsyScope works on all Macintoshes including the MacPlus and PowerBooks. Although experiments can be run on any of these machines, slower CPUs have a hard time keeping up with the computational demands imposed by use of the graphic environment, and by the construction of trials at run time. The graphic environment can be bypassed, by working strictly within the scripting language. However, the time it takes to construct trials (particularly with complex designs) can impose long precompile times (in precompile mode) or intertrial intervals (in trial-by-trial mode). We have found that all around performance is acceptable on a Macintosh IIsi or better.

Input and Output Devices. As noted above, PsyScope supports the use of 1 to 24 bit color, multiple monitors, 8 bit sound in SND resource or SoundEdit formats, and 16 bit sound when used with a Digidesign board. PsyScope recognizes input from the keyboard, the mouse, and the CMU button box (described above). An extension is currently being developed for oscilloscope display (permitting 1 ms resolution for simple visually presented stimuli).

Limitations of PsyScope

Two major types of experiments are currently not well supported. Psycholinguistic experiments involving the construction of sentences from lists of stimuli can be built through PsyScript, but currently cannot be designed in the graphic environment. We plan to eliminate this restriction in the next major version. Second, PsyScope is not well adapted for the design of complex user interfaces involved in studies of Problem-Solving or Computer Assisted Instruction. For such interfaces, it is better to think in terms of using a program like HyperCard.

At present, there is a practical limitation to the size of visual displays that can be presented within a single refresh, using standard Macintosh screens. The severity of this limitation is determined by the speed of the CPU and RAM on which PsyScope is run.

Figure 30

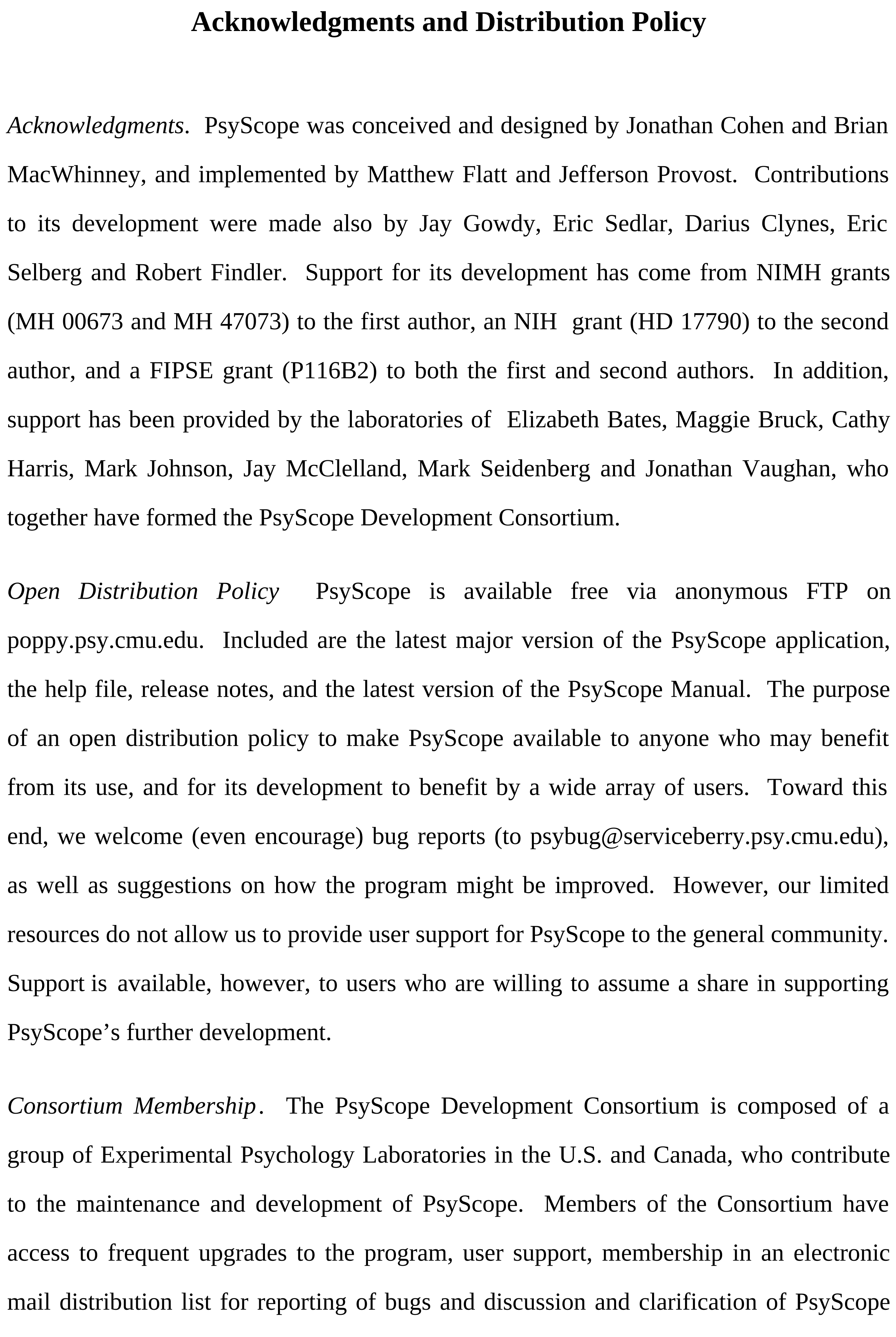
This limitation is due to the use of the standard Macintosh Toolbox call for copying bitmaps from off-screen buffers into display memory. We are currently exploring options for circumventing this limitation (or trading it off against others, such as image complexity), and will incorporate these into future versions of PsyScope.

As noted above, the temporal resoultion of PsyScope is determined by the characteristics of the timing and output devices that are used. Standard Macintosh systems provide a temporal resoultion of 16 ms. With the CMU button box, event and response timing can be improved to 1 ms. Improvements in the temporal resolution of visual displays beyond 16 ms require specialized hardware, for which extensions to PsyScope are currently being developed.

Conclusions and Summary

In summary, PsyScope offers an extensive, integrated environment for experimental design on the Macintosh. It takes full advantage of the Macintosh's graphic environment, allowing novice and experienced psychologists alike to design and implement psychology experiments without any need for programming under most circumstances. At the same time, it offers a powerful scripting language for more specialized applications. Interactivity between the graphic and scripting environments allows novices to discover how the scripting language works, while it allows more sophisticated users to quickly outline an experiment before customizing the script. Furthermore, it provides tools for experienced programmers to extend its capabilities. Most importantly, however, PsyScope is structured around the basic concepts of experimental design, providing an environment in which these can be easily learned by students and used intuitively by experienced psychologists.

Figure 31



Acknowledgments and Distribution Policy

Acknowledgments. PsyScope was conceived and designed by Jonathan Cohen and Brian MacWhinney, and implemented by Matthew Flatt and Jefferson Provost. Contributions to its development were made also by Jay Gowdy, Eric Sedlar, Darius Clynes, Eric Selberg and Robert Findler. Support for its development has come from NIMH grants (MH 00673 and MH 47073) to the first author, an NIH grant (HD 17790) to the second author, and a FIPSE grant (P116B2) to both the first and second authors. In addition, support has been provided by the laboratories of Elizabeth Bates, Maggie Bruck, Cathy Harris, Mark Johnson, Jay McClelland, Mark Seidenberg and Jonathan Vaughan, who together have formed the PsyScope Development Consortium.

Open Distribution Policy PsyScope is available free via anonymous FTP on poppy.psy.cmu.edu. Included are the latest major version of the PsyScope application, the help file, release notes, and the latest version of the PsyScope Manual. The purpose of an open distribution policy to make PsyScope available to anyone who may benefit from its use, and for its development to benefit by a wide array of users. Toward this end, we welcome (even encourage) bug reports (to psybug@serviceberry.psy.cmu.edu), as well as suggestions on how the program might be improved. However, our limited resources do not allow us to provide user support for PsyScope to the general community. Support is available, however, to users who are willing to assume a share in supporting PsyScope's further development.

Consortium Membership. The PsyScope Development Consortium is composed of a group of Experimental Psychology Laboratories in the U.S. and Canada, who contribute to the maintenance and development of PsyScope. Members of the Consortium have access to frequent upgrades to the program, user support, membership in an electronic mail distribution list for reporting of bugs and discussion and clarification of PsyScope

Figure 32

issues. Bugs reported by Consortium members receive high priority, especially if they interfere with the running of experiments. Members also have a voice in directing future PsyScope development and, of course, the good feeling of contributing to a product that is furthering their research. To become a member of PsyScope Development Consortium, please contact either Jonathan Cohen (jc5e+@andrew.cmu.edu) or Brian MacWhinney (brian+@andrew.cmu.edu).

Figure 33

References

Schneider, W. (1988). Micro Experimental Laboratory: An integrated system for IBM PC compatibles. *Behavior Research Methods, Instruments, and Computers, 20,* 206-217.

Vaughan, J. (1992). The dimensions of computing. *Behavior Research Methods, Instruments, and Computers*, *24*, *2*, 109-115.

Figure 34