

Designing Multimedia Internet Experiments  
using Authorware Attain

DRAFT COPY: COMMENTS WELCOME

Alan D. J. Cooke  
University of Florida

Address correspondence to:  
Alan Cooke  
Department of Marketing  
212 Bryan Hall  
P.O. Box 117155  
University of Florida  
Gainesville, FL 32611-7155  
Phone: (352) 392-0161 ext. 1426  
E-mail: [cookea@dale.cba.ufl.edu](mailto:cookea@dale.cba.ufl.edu)

Abstract

Behavioral researchers increasingly require software for designing and conducting Internet experiments. One software package that Internet experimenters are increasingly turning to is Authorware Attain. Authorware Attain provides for the presentation of text, images, sounds, and animation and allows the user to interact with the experiment in complex ways. Experiments can be run remotely over the Internet (or over an Intranet) from a web browser. Communication with the experiment server is possible using a variety of protocols, including HTTP and FTP. I outline the design of Authorware experiments in general and discuss issues related to running Authorware Attain experiments over the Internet. I also provide a compendium of resources for the Authorware designer.

Behavioral researchers have shown increasing interest in using the Internet to collect data. It is hardly surprising that researchers are tempted by the idea of laboratories without walls. Some researchers do not have ready access to the subject populations needed for their research, or require populations that are distributed over wide geographic areas. Other researchers would like to make research participation easier for the subject. Still others are interested in maintaining a laboratory of less-powerful computers that connect to an experiment server via an Intranet or the Internet.

Although many issues related to Internet research have yet to be resolved (c.f. Schmidt, 1997; Birnbaum, in press), the technology for running highly complicated, multimodal experiments over the world-wide-web already exists. In this paper, I will review one software package, Authorware Attain, that is designed for the presentation of multimodal stimuli and the collection of behavioral responses over the Internet. I will provide a brief overview of the software and discuss the issues involved in using it for Internet-based behavioral research.

A variety of tools are available for designing computerized experiments. These tools typically vary in ease-of-use and flexibility. Some software packages are developed specifically for experimenters. For example, E-Prime (<http://www.pstnet.com/e-prime.htm>) allows experimenters to quickly design simple experiments and includes procedures for data analysis. However, it is often difficult or impossible to design experiments having complex designs or involving specialized interaction with the subject using such specialized software. Furthermore, it is not possible to run experiments over the Internet with such software.

On the other end of the spectrum are programming languages like C, Basic, and Java. These programming languages offer great flexibility, but require substantial training. While it is possible to design experiments that can be run over the Internet with these languages, the process is often quite involved and time-consuming. (See Hecht, et al. 1999 and Stevenson, Francis & Kim, 1999 for examples of Internet experiments using the Java programming language.)

Authorware, originally developed for the design and deployment of computer-based training and testing applications, is designed for the middle of this spectrum. Authorware offers a graphical, drag-and-drop interface for the scripting of routine tasks such as presenting stimuli or collecting responses. Authorware also includes a programming language complete with arrays, custom variables, and functions. Authorware has built in various means for communicating across a network and distributes a free Authorware Web Player (formerly called Shockwave for Authorware) that enables subjects to run Authorware experiments on most web browsers. Finally, there is a thriving community of Authorware developers who distribute libraries to perform common or useful tasks.

### **Authoring an Experiment: The Basics**

Wolfe (1992) reviewed many of the basics of Authorware Professional, an earlier version of the Authorware software.<sup>1</sup> His review focused primarily on using Authorware to design courseware, but many of the issues apply to experiment design. Authorware Attain is similar to Authorware Professional, but many important features have been added. I will first briefly review the basics of the software, noting additions.

---

<sup>1</sup> Authorware Attain is the brand name for version 5.x of the Authorware package. Wolfe (1992) reviewed version 1.0.

Experimenters design Authorware experiments by placing icons of different types on the program “flowline”. Available icons are summarized in Table 1. Icons control the presentation of stimuli (either text, images, sounds, or animations), the actions of the program (the order in which stimuli are displayed and erased, and how variables are computed), and how the user can interact with the program (typing, pressing buttons, sliding sliders, and so forth).

Table 1: Authorware Attain icons and their functions.

Authorware Icon	Function
Display	Presentation of static images, movable images, drawings, text, or variables.
Motion	Moves a screen object along a predetermined path.
Erase	Erases the contents of one or more display icons.
Wait	Pauses the program until a user event or for a fixed amount of time.
Navigate	Directs program flow to a predetermined or subject-selected icon.
Framework	Contains a hypermedia structure of other functions that subjects can navigate through and search.
Decision	Allows the program to execute different paths depending on the subjects’ responses.
Interaction	Allows the subject to react to stimuli by pressing keys, entering text, clicking buttons, etc.
Calculation	Calculates variables and executes various functions.
Map	Groups sets of icons into one icon.
Digital Movie	Presents digital movies.
Sound	Presents prerecorded sounds.
Video	Includes still images, sound, or animation from videodisc.
Knowledge Object	Allow the scripting of design tasks with a “wizard” interface. Experimenters enter parameters that modify the actions of the underlying icons.

### *Birds and Planes: A Simple Experiment*

As an example, I describe a simple two-condition experiment. The experiment randomly assigns subjects to one of two conditions. Subjects in the first condition see a picture of a bird and the word “bird”. Subjects in the second condition see a picture of a

plane and the word “plane”. Subjects select either the animate or inanimate button, depending on the stimulus. The condition and response are then written to a text file.

Figure 1 shows the Authorware file for this experiment. The first icon (“Condition”) is a decision icon. Double-clicking the icon will open a dialog box of icon properties including options for branching and repeating. We want each subject to only receive a stimulus once, so we set the properties to “Don’t Repeat” and “Branch Randomly to Any Path”.<sup>2</sup> This will cause the program to select one of the two attached map icons (“Birds” or “Planes”) randomly and begin executing that map icon.

-----  
 Insert Figure 1 about here.  
 -----

Each of these map icons includes four icons, a display icon, an interaction icon, and two attached calculation icons. The display icon displays the stimulus. Double-clicking this icon will open the display. Text can be added by using the text tool. (A variety of other simple drawing tools are included.) The picture can be added by using the “insert | image” menu listing. Double-clicking the interaction icon will also open the display. This allows experimenters to enter text or pictures that should only be displayed when the interaction begins, for example, the text of the response question. The interaction icon points to two ovals indicating button interactions.

When the subject presses one of the buttons, the code in the corresponding calculation icon is executed. In our case, the code includes three commands:

```
Response := "animate"
```

---

<sup>2</sup> To run the experiment within subjects, we would set the properties to “Repeat Until All Paths are Used” and “Branch Randomly to Unused Path”.

```
Str:="Birds = "^Response^Return  
AppendExtFile(FileLocation^"data.txt",Str)
```

The first line sets the variable “Response” equal to “animate”. The second line appends this response to a string specifying the condition. (“^” is the concatenation operator, and “Return” specifies a newline character.) The third appends this string to the file data.txt in the folder pointed to by the FileLocation variable. FileLocation is a system variable that contains the folder from which the program resides. The icons enclosed in the “Planes” map icon are the same as in the “Birds” map icon, except that the display icon displays a plane and the calculation icons specify the planes condition.

#### *More Birds and Planes: A More Realistic Experiment*

Suppose that you wanted to run a more complicated experiment. In particular, suppose that you want to present subjects with a series of different pictures of birds and planes in random order. Again, you want subjects to respond by reporting whether the object pictured is animate or inanimate. However, in this case you want to record not only the response but also the reaction time. Furthermore, you want to manipulate which response label is assigned to each response button between subjects.

Luckily, this design requires only minor modification of the preceding experimental program. First, we must select the response condition. We first present a screen in which the experimenter can select one of two response conditions. This sets the variable RespCond, which controls the button position later in the piece.

Now, we need to present the ten images in random order. For each trial, we need to present an image, collect a response, and write it to a file. We will enclose these

---

functions inside a decision icon that is set to repeat a fixed number of times (ten). One simple approach for displaying the different images is to place each image in a separate display icon and attach all these display icons to a single decision icon, as shown in Figure 2. The decision icon is set to repeat until all paths are used and to branch randomly to an unused path.

-----  
Insert Figure 2 about here.  
-----

This approach is reasonable for relatively small stimulus sets, but is clearly wasteful for large designs. A more economical approach (at least in terms of the number of display icons one must use) is shown in Figure 3. In this program, we create a randomly ordered stimulus index and use this index to link to externally stored images. The code shown in the appendix (taken from the calculation icon “get random index”) will generate a randomly shuffled index from 1 to 10. We place a single display icon on the flowline and display an image having the name “imageX” where X is the value of the index. We do this by specifying “image”<sup>index[trial]</sup> for “file” in the image properties dialog box.

-----  
Insert Figure 3 about here.  
-----

Next, we present one of the two response conditions by branching from a decision icon. The decision icon is set to not repeat and to branch to a calculated path based on `respond`. We then write the trial number, index, response, and response latency to an

external file in a calculation icon. The system variable `TimeInInteraction` contains the response latency for the most recent interaction icon. Now we just repeat the trial.

### **Deploying your Experiment**

Now that you have a working experiment, you want to collect data. First, you must package your experiment. You can package Authorware programs with or without the Authorware runtime engine. When packaged with the runtime engine, the program is compiled into executable code that can run on any 16 or 32-bit Windows machine. Now you need merely copy the executable file, along with any external media files and Xtras it uses to each machine in your laboratory.

If you wish to run your experiment over the world-wide-web, you must first package the program *without* runtime. Then, you run the Authorware Web Packager on the resulting file. This divides the program into segments that are downloaded individually by the Authorware Web Player running on the subject's computer.

Authorware Attain includes new functionality for intelligent content streaming which downloads segments in the background prior to when they are needed, and downloads only those segments that are needed. This means that large, stimulus-rich experiments can be downloaded over low-bandwidth connections without long delays.

Subjects who want to run the experiment over the Internet will need to install the Authorware Web Player. This software is free and is available at

<http://www.macromedia.com>. The installation file is quite large (approximately 4 MB) which

may deter some potential subjects from downloading it. However, this concern is not as relevant when running experiments from existing laboratory computers.

### **Security and the Internet**

Authorware Attain includes functions that enable the experiment to write data to the user's hard drive, set permissions for files and directories, and delete files from the user's hard drive. As such, Authorware programs represent a potential security risk to the subjects who wish to run them. To limit this risk, Authorware includes security measures invoked when subjects access an experiment using the Authorware Web Player.

Experimenters using Authorware over the Internet should familiarize themselves with these measures and take them into consideration when designing an Internet laboratory.

Programs run using the Authorware Web Player may be run in one of two security modes. When run in non-trusting mode, the web player disables potentially dangerous variables (such as those that report information about the subject's system and directory information) and functions (such as those related to the creation and removal of files and directories, and the execution of other programs). Furthermore, in non-trusting mode, all access to external content is disabled. When run in trusting mode, the piece is permitted to use these variables and functions.

Web player security is invoked as soon as a user attempts to run the experiment. The subject's system will first display a dialog box asking the subject whether or not to trust the piece. Subjects can trust individual experiments, or can adopt the trusting mode for all experiments on a particular server or directory. (The security dialog can be bypassed for programs designed to be run in non-trusting mode.) Experimenters can also

edit and distribute the file `AWSHKWV.INI` which can be configured to automatically trust particular sites. This solution is particularly useful when running networked experiments over designated machines.

### **Authorware Attain Communication Techniques**

The Authorware Web Player allows subjects to play Authorware experiments over the World Wide Web. This will typically suffice when one wishes to demonstrate behavioral concepts or provide class tutorials. However, to collect data over the Web, you must be able to communicate the subject's responses back to the server. Authorware Attain includes a number of functions related to Internet communication, either internally or through Xtras. Table 2 summarizes selected communication functions that are built into Authorware Attain.

Table 2: Selected system communication functions

Function Name	Description
NetDownload	Downloads a file specified by a URL to the local drive and returns the path and file name.
NetDownloadBackground	Downloads a file in the background. Can report progress and errors, and can be aborted.
PostURL	Posts content to a specified URL and returns the result. Can be used to communicate with CGI scripts.
ReadURL	Reads a specified URL and returns the file's contents or the JavaScript result to a string.

Three technologies exist within Authorware Attain for communicating a subject's responses back to a server. The first utilizes a built-in (system) function named `PostURL`. `PostURL` posts the content of a string to a specified URL, such as a CGI script. Although

this approach is the easiest with respect to writing the Authorware program, it requires that the experiment designer be adept at writing CGI scripts or have in place a fairly general CGI script for accepting data.

An alternative means of communication involves use of the FTP UCD. FTP stands for File Transfer Protocol, an asynchronous communications protocol developed specifically for the transfer of files between computers. The FTP UCD contains a variety of functions that enable the Authorware program to open a connection to an FTP server, upload or download files or directory listings, and close the session. The most useful FTP functions are summarized in Table 3.

Table 3: Selected functions in the FTP UCD

Function Name	Description
FtpOpen	Initializes an FTP session.
FtpConnect	Connects to an FTP server.
FtpRetrieve	Gets a single file by FTP.
FtpList	Gets multiple files from a remote machine by FTP.
FtpStore	Writes a file to a remote machine.
FtpAppend	Appends a file to a remote machine.
FtpResult	Returns the result of an FTP function.
FtpDisconnect	Disconnects from an FTP server.
FtpClose	Closes an existing FTP session.

To transfer data using FTP, you will need an FTP server running on your host machine. Microsoft's Internet Information Server (IIS) includes an FTP server, and many other shareware FTP servers are available online. One advantage of using FTP to communicate (as opposed to posting the data to a CGI script) is that the communication is handled within your Authorware piece, so the piece is able to easily detect and report communication errors and act accordingly. A second advantage is that the approach is general: as long as the program knows what server to contact and what files to transfer,

the same code may be used in any number of experiments. An example Authorware piece for uploading files is available at <http://weblab.cba.ufl.edu/weblab/>.

A third approach for sending data back to the experimenter uses Open Database Connectivity (ODBC). Authorware includes an ODBC UCD that allows the experimenter to store, retrieve, and modify data in many ODBC compliant databases (including MS Access, dbase, Oracle and Paradox) using a language called the Structured Query Language (SQL). Table 4 lists the ODBC functions available in the UCD. There is a function to open the connection, one to execute a SQL query, and a third to close the connection.

Table 4: Functions in the ODBC UCD

Function Name	Description
ODBCOpen	Opens a session with an ODBC database and returns a database handle.
ODBCExecute	Executes the SQL commands in a specified string and returns the result.
ODBCClose	Closes the ODBC session.

Using ODBC to communicate requires that the experimenter install an ODBC compliant database and develop a working knowledge of SQL. (Many excellent books and on-line SQL tutorials have been written. A good introduction to SQL is available at <http://w3.one.net/~jhoffman/sqltut.htm>) Furthermore, because the structure of the data will change with each experiment, the experimenter will need to make frequent changes to the data tables. For this reason, I do not recommend this approach for developing small Internet based laboratories having many different experiments. However, ODBC is recommended for the deployment of Internet surveys or experiments where many subjects are expected to participate in a small number of unchanging studies. In fact, for

large-scale studies, storing the data directly to an on-line database has the added benefit of allowing researchers around the world to access and explore the data as they arrive.

Many other sciences are moving in this direction, most notably astronomy.

In sum, there are three sets of techniques that can be used in Authorware to communicate data back to the experimenter. Posting the data to a CGI script requires the least programming in Authorware, but requires the development of CGI scripts.

Transferring subject files by FTP is useful for Internet based laboratories with multiple experiments, but does require that the experimenter install and monitor an FTP server.

Storing the data to an ODBC database is useful for large, unchanging studies, but involves considerable learning costs.

### **Conclusion**

Authorware Attain is an extremely versatile package for the development of computerized experiments. In particular, Authorware allows experiments to be run over the Internet (or a LAN) via popular Web browsers. Authorware includes functions related to the security of the piece, the intelligent streaming of data to the client's machine, and the communication of results to the experimenter.

A variety of communication techniques exist: Data can be posted to a CGI script, as is typically done with HTML forms. If subjects' responses are stored to a local file, this file can be sent to the host machine via FTP. Finally, data can be stored to and retrieved from databases using ODBC. Each of these techniques may be useful depending on the skills of the experimenter, the scale of the project, the diversity of studies involved, and the experimenter's plans for disseminating the data.

As noted by Wolfe (1992), learning to use Authorware requires a substantial time investment. Aspects of the software have become easier to use since the time of Wolfe's review, but the software has also become considerably more complex. This complexity is not particularly beneficial for those designing simple computerized experiments to be run on standalone computers. However, for experimenters interested in presenting multimedia stimuli, allowing for non-standard subject interactions, or wishing to conduct the experiments over a network, Authorware provides a powerful and flexible solution. As the popularity of the software grows among behavioral researchers, functions tailored to the experimental community will likely become available. These developments bring behavioral research one step closer to the dream of a laboratory without walls.

## References

- Birnbaum, M. H. (ed.) (in press) *Psychological Experiments on the Internet*. San Diego, CA: Academic Press, Inc.
- Hecht, H., Oesker, M., Kaiser, A., Civelek, H., & Stecker, T. (1999). A perception experiment with time-critical graphics animation on the World-Wide Web. *Behavior Research Methods, Instruments, & Computers*. **31** (3), 439-445.
- Schmidt, W. C. (1997). World-Wide Web survey research: Benefits, potential problems, and solutions. *Behavior Research Methods, Instruments, & Computers*. **29** (2), 274-279.
- Stevenson, A. K., Francis, G., & Kim, H. (1999). Java experiments for introductory cognitive psychology courses. *Behavior Research Methods, Instruments, & Computers*. **31** (1), 99-106.
- Wolfe, C. (1992). Using Authorware Professional for developing courseware. *Behavior Research Methods, Instruments, & Computers*, **24** (2), 273-276.

## **Appendix: A selective compendium of Authorware resources**

On-line documentation:

- Macromedia's Developers Center:

<http://www.macromedia.com/support/authorware/attain/documentation/Authorware>

Authorware tutorials and FAQ lists:

- Macromedia's Developers Center:

[http://www.macromedia.com/support/authorware/attain/dev\\_index.fhtml](http://www.macromedia.com/support/authorware/attain/dev_index.fhtml)

- Authorware beginners' site: <http://home.sprintmail.com/~possumhome/>
- The Media Shoppe: <http://www.mediashoppe.com/files/>
- Authorware Knowledge Base: <http://www.kw1c.nl/Aware/index.htm>

Sources for Xtras, UCDs, and sample files:

- Successful Multimedia Inc.: <http://www.mrmultimedia.com/indexnoflash.htm>
- The Media Shoppe: <http://www.mediashoppe.com/files/>
- The Authorware Resource Center: <http://www.stingray-interactive.com/awunder/index.htm>
- Magic Modules: <http://www.mods.com.au/>

Electronic mailing lists:

- The Authorware Mailing List: <http://www.e-media.nl/aware/>

Webrings:

- The Authorware Webring: <http://www.webring.org/cgi-bin/webring?ring=authoring:index>

## Figure Captions

Figure 1: Authorware Attain program for a two-group between-subjects manipulation.

(All example programs are available at <http://bear.cba.ufl.edu/cooke/labbench/files/awareex.zip>.)

Figure 2: Authorware Attain program for a repeated-measures design.

Figure 3: The same experiment as in Figure 2, using a single display path.

## Appendix

This appendix discusses the Authorware code used to generate a randomly shuffled stimulus index in Figure 3.

```

1  --make the array of image names
2  Stim=["bird1.jpg", "bird2.jpg", "bird3.jpg",
        "bird4.jpg", "bird5.jpg", "plane1.jpg", "plane2.jpg",
        "plane3.jpg", "plane4.jpg", "plane5.jpg"]
3  NStim:=10
4  --make an array of integers
5  i:=1
6  --populate the index
7  repeat while i<=NStim
8      AddLinear(index,i,i)
9      i:=i+1
10 end repeat
11 --shuffle the index
12 i:=1
13 repeat while i<=NStim
14     --calculate a random offset into the index
15     j:=Random(1,NStim-i+1,1)
16     --swap the ith value with the i+jth value
17     Temp:=ValueAtIndex(index,i)
18     SetAtIndex(index,ValueAtIndex(index,i+j-1),i)
19     SetAtIndex(index,Temp,i+j-1)
20     i:=i+1
21 end repeat

```

Lines beginning with two dashes (lines 1, 4, 6, 11, 14, and 16) are comments and are not interpreted by Authorware. The second line declares a linear list of ten elements, where each element is a string naming one of the image files. Line 3 sets the variable NStim to 10. Lines 5 through 10 use the AddLinear system function to insert the

number  $i$  in the  $i$ th position of the list `index` for  $i$  from 1 to 10. The next loop (lines 12 through 21) iteratively swap each value in `index` with a randomly selected value from the end of the list. Line 15 uses the `Random` function to generate a random integer,  $j$ , between 1 and `NStim-i+1`. Lines 17 through 19 swap the  $i$ th value of `index` with the  $i+j$ th value of `index`, using the system function `SetAtIndex`. Line 20 iterates  $i$  and the process is repeated until the end of `index` is reached.

Figures

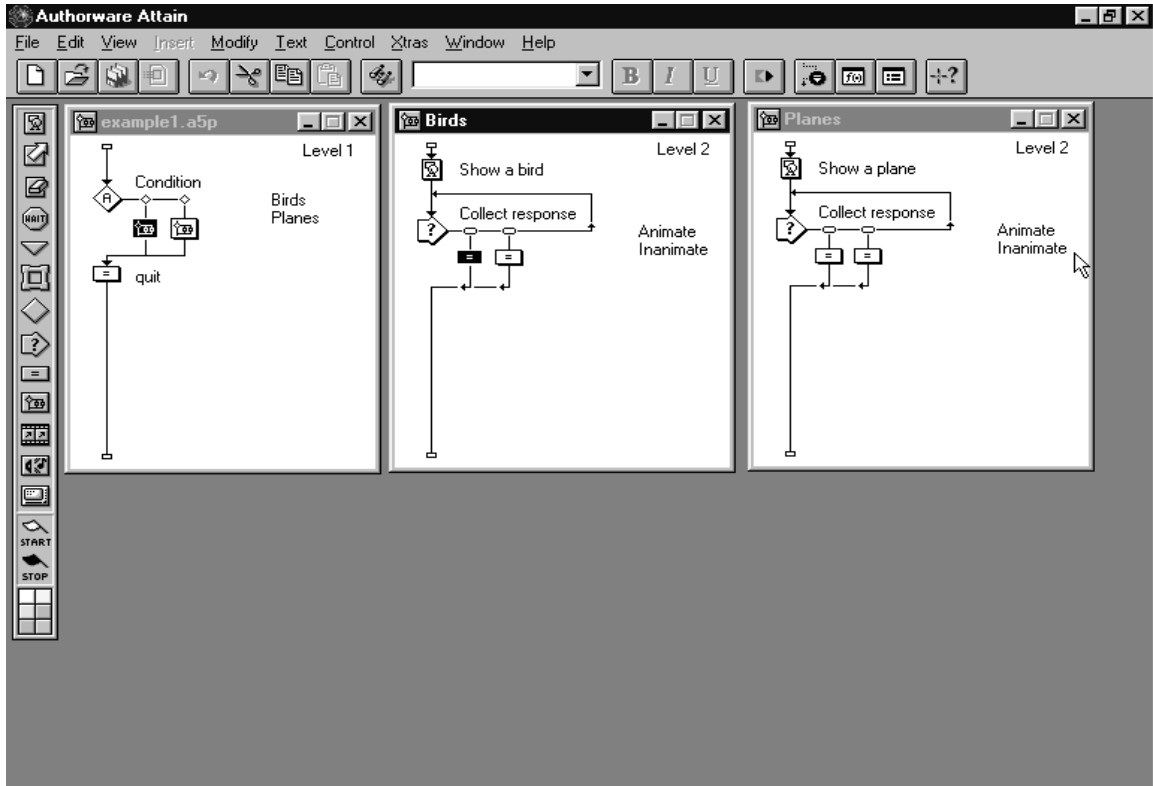


Figure 1

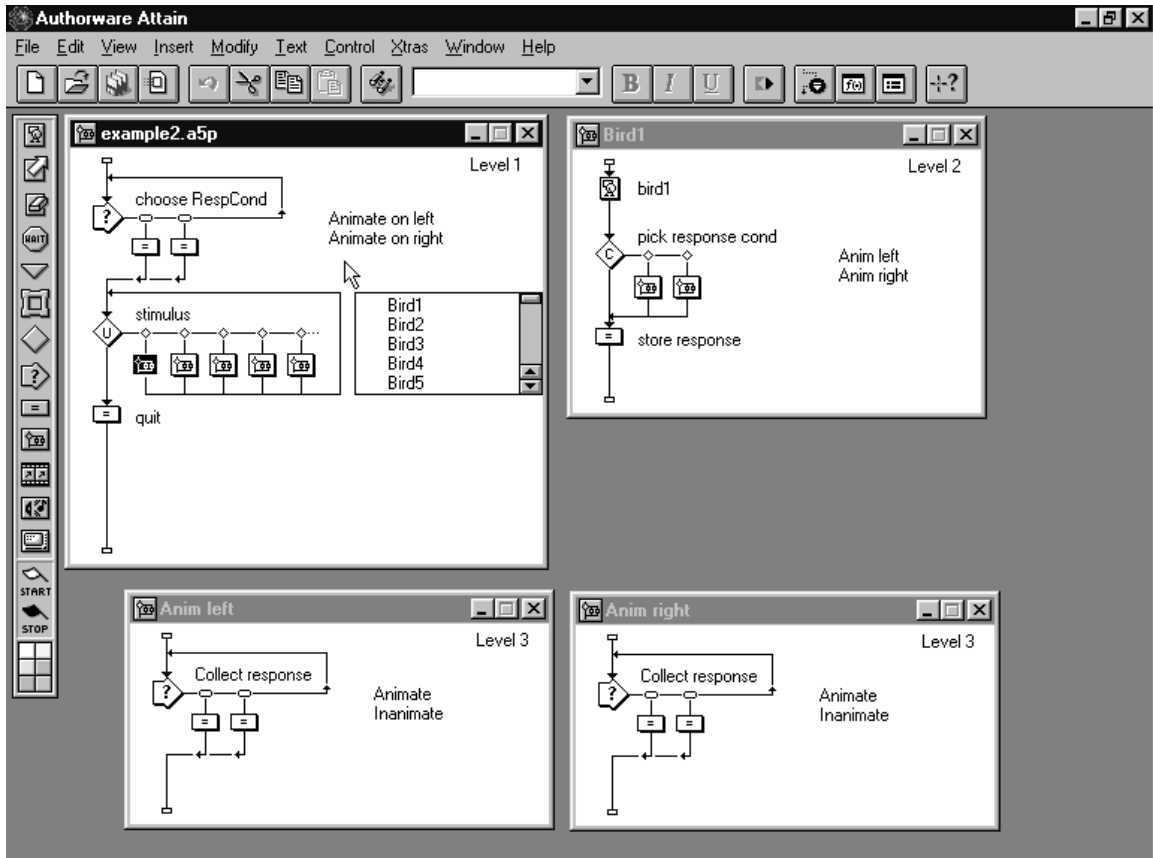


Figure 2

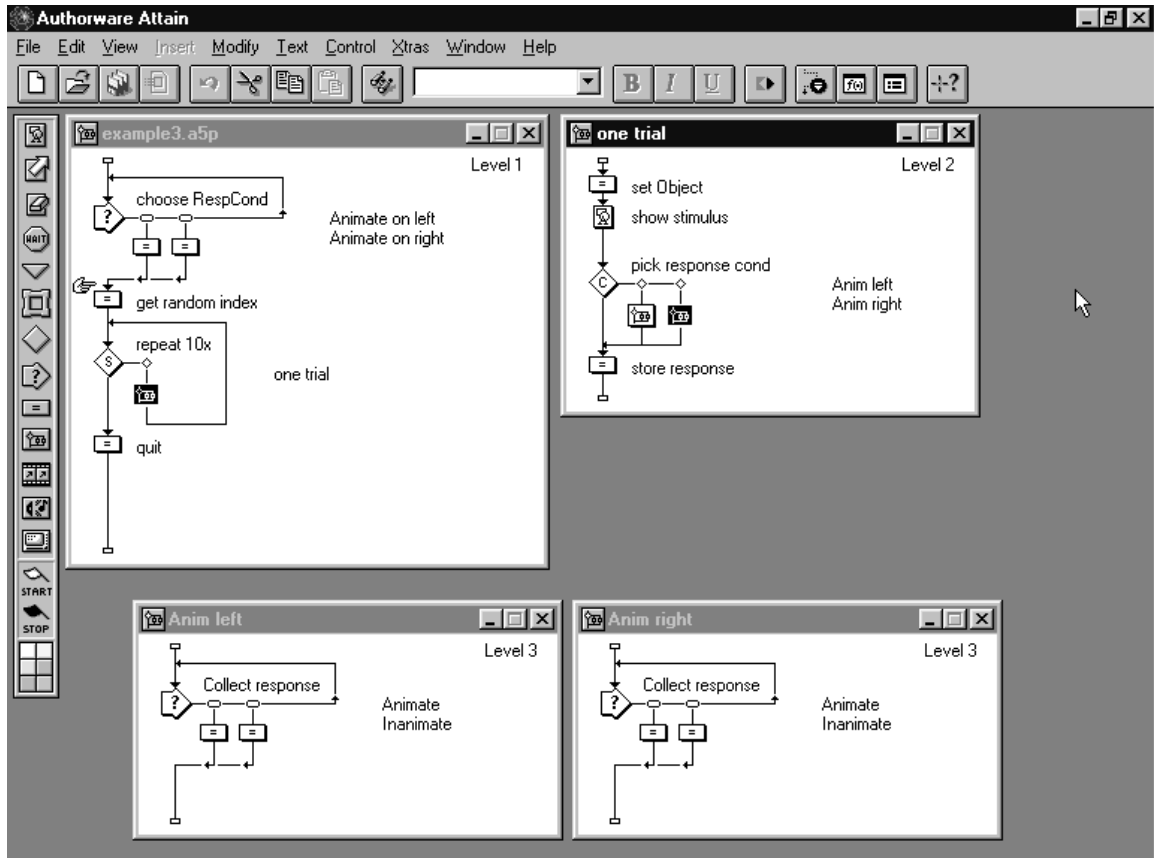


Figure 3