An Auditory and Visual Experimental Control System

Luís R. Hernández and Caleb R. Marcinkovich

Speech Research Laboratory
Department of Psychology
Indiana University
Bloomington, Indiana 47405

1 The development of this system was supported by NIH-NIDCD Research Grant DC00111 to Indiana University-Bloomington, IN. The authors would like to thank Jim Sawusch for his contribution and assistance in developing this software.
An Auditory and Visual Experimental Control System

Abstract. This report describes the development of a new multi-modal auditory and visual (A+V) testing experiment system. We outline the hardware requirements and describe the implementation of a system that can present full-motion video and record accurate reaction times from subjects responses. A Macintosh equipped with a video board and an analog input/output board was selected as the preferred platform. A set of general purpose high-level library calls were written to simplify the implementation of future experimental control programs.

Overview

The goal of the project was to design and implement a set of experimental perceptual paradigms that would present digital audio and video cross-modal stimuli. Some examples of the paradigms that we had in mind are identification, discrimination, recognition memory, recall and lexical decision tasks. Previous experiments have been carried out using video tape and in some instances Laserdiscs. However, this traditional analog media limits control over stimulus presentation and accuracy over latency measures. Furthermore, we wished to assemble a collection of our own movie-playing, randomization and event-timing routines as a high-level library set to provide other programmers convenient tools that would facilitate future development. An experimenter’s ability to precisely control the onset and offset of stimuli presentation, as well as, control over timing between events is crucial to the success of any experiment. All stimulus events are recorded on a log file.

Stimuli consist of a full-motion video of a face articulating words or sentences. These stimuli could be edited and modified according to the needs of the particular experiment. We are planning to develop the ability to edit and modify the audio and the video tracks of the digitized movies independently of each other, maintaining the ability to synchronize the tracks. For this report, we will not discuss how the stimuli are digitized and edited, but will refer to stimuli as files that consist of segmented preprocessed movies of different duration varying from two to six seconds.

Design Criteria

With these goals in mind, we have decided on a number of design criteria. The system should rely, as much as possible, on standard, off-the-shelf components so that all users may take advantage of competitive pricing as well as simplified maintenance. The system should also be designed in such a way that it can be implemented inexpensively for simple, frequently-used experimental paradigms in our research program. However, as the sophistication and resolution of the experiments increases, the cost of additional equipment will increase as well. The system should be easy to maintain and manage, preferably benefiting from a graphical user interface (GUI). Although it is not our immediate goal, software should be as transportable as possible, and hardware components should be available in different platforms.

Functional Requirements

The principal function of our system is to play full-screen, full-motion video at NTSC standards. This usually means 30 frames per second (fps) at a resolution of 24-bit color on a 640 by 480 resolution monitor. We would like to display the stimuli as close to life size as possible while maintaining the resolution desired. The video image needs to be computer-controlled as precisely as possible in order to
capture millisecond accuracy between any two chosen events. The audio requirements are 16-bit resolution with sampling rates up to 44.1kHz synchronized with the video track. Given that the video is playing at 33.3 msec cycles, we would expect the latencies to be only this accurate. But we would still want to maintain a one millisecond resolution for audio-alone experiments and other non-video events.

The types of subject responses that the system needs to handle are: (1) handwritten, (2) keyboard, and (3) button-presses. For handwritten input, the system needs only to play the video at a fixed inter-stimulus interval (ISI) specified by the experimenter. Keyboard input constitutes a string response prompt with a time-out period after the stimuli are presented. Finally, a button-press response needs to capture which button was pressed and accurately measure time between certain events (e.g., stimulus onset to button press). The response boxes also need to have feedback lights above each button, as well as a cue light in the middle top portion of the box. Button responses will be signaled on the down press only, with a short travel distance. Button size is not critical, but will be consistent. For experiments in which latency is measured, keyboard responses will not be acceptable due to small, unpredictable variations in response time. Moreover, mouse responses, voice-operated switches, and touch-screen inputs will never be used to measure latency. A log file needs to record randomization order, current trial, key of correct responses, subject’s response, and response latency. Each subject will have control over the pace of the experiment and perform the task independent of other subjects.

Modular software design is essential to ensure well-written code, ease of support, ease of maintenance, portability and reusability of our libraries. The software library will have at a minimum, capabilities analogous to those of the existing Perceptual Testing System (PTS) calls (see Hernández, 1995). However, the routines will require extended capabilities given that they do not support video.

**System Configuration and Components**

We chose the PowerPC 8100 Macintosh platform for the development and implementation of these applications because of Apple’s long history of robust A+V capabilities. The system would benefit from the inherited designed GUI providing an easily navigable setup-and-run environment. 36MB of RAM provides enough memory to maintain the throughput required. Since compressed digitized movies of this resolution require approximately 4MB (27MB uncompressed) of storage space per second, we have equipped this computer with a 5 gigabytes (GB) hard disk. A 17-inch Trinitron tube non-interlace monitor was chosen for the display.

To achieve real-time video capable of playing full-screen QuickTime movies, we decided on the Radius VideoVision board. The Radius board provides flicker-free, smooth playback for full-screen, playing 30 frames per second (fps), 24-bit color movies using special compression and decompression (codec) algorithm. This particular board can play and record.

The input/output (I/O) component utilizes the Strawberry Tree ACM2-I/O analog I/O card for timed button box response. This board contains 40 I/O lines and six counter/timers. We utilize two of the six hardware counters to produce millisecond accuracy for response latency. The timer routines were provided by James Savusuch at the State University of New York at Buffalo. The first 12 of the 40 digital I/O lines are currently dedicated to button response and feedback lamps. The button-response boxes were developed by the Indiana University Psychology Department Electronic Shop. Detailed description of what is needed to connect from the boxes to the I/O card can be found in the Strawberry Tree manuals. For most experiments, we will use two- or seven-button response boxes with three or eight feedback lamps.
However, the button-box routines are capable of employing response boxes with anywhere from one to eight buttons and feedback lamps.

We picked Metrowerks CodeWarrior Gold version 5.0 for its capability to compile and generate executables for different platforms and for its appealing debugger. We complemented the C compiler with Apple’s QuickTime2.0 developers kit.

**Implementation Phases**

**Phase 1: Handwritten Response**

We usually set out to develop routines and put together systems with a particular set of experimental paradigms in mind. In this case, however, we started by implementing a basic identification task (MovieID). For this first phase, the application would merely play movies with a fixed ISI. A skeleton of code from a movie controller provided by QuickTime SDK was taken as a model and standard C timing routines were used. This application would operate in the following steps: First, a stimulus-set-file (SSF) is created. Next, all experimental settings are typed or modified from the default, and finally the experiment runs, and timestamps are logged to a file.

The SSF is a text file containing the list of filenames of the movies to be played. The user creates this file before hand using a general text editor (entering the name of each movie per line). MovieID uses this file to read in the names of the movies to be played. The Mac simplifies the process of typing in a long list of names by its copy-and-paste (i.e., copying from the desktop and pasting to the SSF text file) desktop editing feature. This file is saved in the same subdirectory where the movies reside.

We developed a user-friendly GUI dialog box that allows for easily modifiable variables which the experimenter can modify to meet the requirements of a particular experiment. These settings are recorded to a log file to maintain an accurate report of experimental setup. The setup dialog box allows the experimenter to modify the following settings:

- General comment
- Log file name
- Cue file name (optional)
- SSF name
- Number of blocks
- Number of repetitions
- Randomization (on/off)
- Inter-stimulus interval time (ISI)
- Inter-trial interval time (ITI)

A set of default values is also available with simple click of the mouse. We realize that for this type of experiment, the accuracy for ISI and ITI is not very important given that 20 seconds are usually allowed for subjects to make their responses (ISI) using ITI times of 5 seconds or more.

To maintain an accurate record of the experiment, the data are logged (recorded) to a file for post-experimental analysis. This file contains a header with all of the modifiable settings. Following the header, the actual experimental data are provided including the trial number, repetition number, token number, and movie filename. The footer of this file contains a timestamp showing the date and time of completion of the
experiment. This file is saved in ASCII text format which can be easily loaded into a spreadsheet (e.g., Excel) or statistical software (e.g., Statview) for analysis.

MovieID presents an option for the presentation of a cue-movie between the presentation of stimulus-movies. The purpose of this cue is to prepare the subjects for the next stimulus. We decided to keep the cue in the format of a movie since this would allow not only a dynamic movie as a cue but also a static-movie (an image), a sound-only movie (only sound output with black screen), or a combination of the two, as most movie-making applications would allow the experimenter to create.

The timeline of the MovieID experiment consists of the sequence; stimulus, ISI, cue, and ITI. If a cue is not present, the ITI time is ignored. During any of the delay-intervals (e.g., ISI and ITI), a blank screen is presented. This run-time sequence is very general and is used in the next two applications with minor modifications. This experiment also has the capability of being prematurely terminated during the presentation of any token or cue.

During the development of this first phase, we realized that we did not have much control over exactly when a movie started playing. Another observation was that the application required at least 30MB of RAM before it could display the movie smoothly and without interruptions. To help the processor maintain its throughput, we turned off all unnecessary extensions. Finally, it will also be helpful to defragment the disk when large movies are played.

**Phase 2: Keyboard Response (MovieIDkb)**

The second step was to modify the first version of the movie-player to accommodate keyboard responses from the subject after the presentation of each movie. All subject responses, consist of a series of keystrokes terminated by a “return.” These are logged to a file as they are input into the computer.

The MovieIDkb has the same functionality as the original MovieID identification experiment but has a few minor changes. The setup dialog box now allows the experimenter to modify the instruction for the dialog box in which the subject enters his/her response. Among the body of experimental data being logged to file, the subject’s responses are also logged to file as their input. The timeline of the MovieIDkb is exactly the same as MovieID except that the subject is prompted to type their response. The subject’s keyboard response is not timed, and therefore has an indefinite period of time to respond.

This particular phase seemed easier to implement that the first, with the exception that we had to worry about insuring that no keyboard combination would interrupt the program. Also, we came across our first non-standard dialog box (one not from the developer’s library) and had to deal more closely with the Macintosh event handler. If this software is to be transportable, it needs to be restricted to the Macintosh family of computers.

**Phase 3: Button Response (Forced Choice Experiment, MovieID-RT)**

For the third phase, we modified the original version of MovieID to accommodate real-time button-box response with feedback. The subject is now forced to make a decision about each stimulus presentation, at which point the button number, latency and current frame would be immediately recorded.

Feedback is also available through the LED sitting on top of each button on the button box. An additional file paired with the SSF is specified in the set-up dialog called stimulus name file (SNF), and is
used to keep track of correct responses. This file contains a list of numbers ranging from 1 through 7, corresponding to response buttons which are labeled accordingly from left to right.

The timeline mimics that of the original version but contains a few modifications to accommodate the paradigm. The response box is deactivated prior to the playback of the token as well as, at any time after a response has been made to avoid recording multiple button responses. If feedback is specified, the correct answer will be illuminated on the button box for a specified time.

In addition to the experimental data being logged to file, the subject’s button response number, response latency, and current movie frame are also logged to file. The response latency and movie frame are both included to provide the experimenter with a precise means of detecting any deviation from actual response time and frame number.

To measure precisely the time from the onset of the stimulus to the button press, we had to load the movie into memory and have the first frame waiting in the hardware’s buffer. With this strategy, the movie would start playing immediately after the “StartPlaying” routine call. These routines are available in the QuickTime libraries, making it possible for us to develop this strategy without having to program at a lower level. The other component needed for accurate measurement is the digital I/O board. The programming of this board required us to work at a register hardware level. Although we had to study and modify some routines to make this work with our system, we are very grateful to Dr. James Sawusch for providing us with timer and response-box routines he previously developed for his own system using the same I/O board.

**Development of Resource Tools**

The final goal of this project was to assemble a group of movie-player and Mac interface routines and provide a general code skeleton for future A+V development on the Mac. Two sets of libraries were developed. One library was created and it contains general routines such as displaying a dialog box, creating a randomized list of numbers, delaying a certain length of time, and converting Mac strings (Pascal-style) to “C” strings. In the past, we have noticed that most of the development time is spent fine-tuning the GUI. We hope that this group of routines will allow more time to be dedicated to the functionality component of development. The second library contains full-screen movie-player routines such as initializing the movie environment, starting and stopping a movie, capturing a frame number, and loading the movie into memory. These routines are the core of functions that allow us to perform the tasks with the necessary flexibility and precision.

As with previous libraries developed in our laboratory over the years (Forshee 1975, 1979; Forshee & Nusbaum, 1984; Hernández, Carrell, Reutter, & Bernacki, 1992), we hope to simplify software development for non-programmers. Giving the users the ability to plan and experiment with their own programs, has yielded a better environment for researchers using computerized experimental paradigms.

In summary, The Macintosh system that we have constructed has proven to work for our initial purposes. We feel that we understand the hardware and software components of the system well enough to have control over the stimulus presentation and latency measures to develop these and other perceptual experiments. We feel that we have met the functional and hardware requirements as described and will continue to test and develop the system as newer paradigms arise.
Discussion

There are some design issues that still need to be discussed and are worth mentioning at this time in development. We have purposely ignored accuracy between events that are not measured with the digital I/O card (i.e., ISI and ITI). This assumption is fine for the paradigms described in this paper, but the addition of precise countdown timers and wait routines are unavoidable. In addition, more external testing should be done to insure that internal and external measures are the same.

Another issue that has not been investigated is the position of the raster when the movie is loaded to the screen. Not only do we need to insure that the raster is positioned at the top left corner of the screen before display, but we also need to measure the time it requires to traverse the entire screen as was done previously for our PTS system (Hernández, 1995). It would be convenient at this stage to calibrate all monitors used in a single experiment with each other to insure uniformity of color parameters.

More testing needs to be done on the audio output to establish a base line signal-to-noise ratio, distortion and other parameters that will inevitably be a factor in many of the experiments conducted here.

Finally, the market changed the Macintosh bus standard during the development of the system described above from NuBus to PCI. Although a Radius VideoVision PCI card is available with the same features as the NuBus card, Strawberry Tree does not manufacture an I/O PCI card equivalent to the ACM2. In order to avoid rewriting the I/O component from another company that offers a PCI I/O card, we will use either older NuBus Macintoshes or newer Macintosh-clones which offer both the PCI and NuBus architectures. Again, further external tests will have to be conducted to determine if the different computer models behave the same way. Also during this same period of development, other companies have developed video cards that can compete with VideoVision, both in terms of performance and, importantly, price. The primary product among these is the Targa board from Truevision. We are seriously considering replacing our VideoVision boards with Targa boards. An extremely important consideration, however, is that movie formats compiled with VideoVision are not compatible with Targa, and vice versa. On the other hand, the only changes to the software would be recompiling with the Targa codec. Currently, we have tested our system only on 950AV Macintosh and PowerPC Macintosh 8100 models.

References


