JSLHR

Tutorial

Phonetics Exercises Using the Alvin Experiment-Control Software

James M. Hillenbrand,[a] Robert T. Gayvert,[b] and Michael J. Clark[a]

Purpose: Exercises are described that were designed to provide practice in phonetic transcription for students taking an introductory phonetics course. The goal was to allow instructors to offload much of the drill that would otherwise need to be covered in class or handled with paper-and-pencil tasks using text rather than speech as input.
Method: The exercises were developed using Alvin, a general-purpose software package for experiment design and control. The simplest exercises help students learn sound–symbol associations. For example, a vowel-transcription exercise presents listeners with consonant–vowel–consonant syllables on each trial; students are asked to choose among buttons labeled with phonetic symbols for 12 vowels. Several word-transcription exercises are included in which students hear a word and are asked to enter a phonetic transcription. Immediate feedback is provided for all of the exercises. An explanation of the methods that are used to create exercises is provided.
Results: Although no formal evaluation was conducted, comments on course evaluations suggest that most students found the exercises to be useful.
Conclusions: Exercises were developed for use in an introductory phonetics course. The exercises can be used in their current form, they can be modified to suit individual needs, or new exercises can be developed.

The purpose of this tutorial is to describe a collection of computer-controlled exercises designed to provide practice in phonetic transcription to students who are taking introductory coursework in phonetics. The main goal was to allow instructors to offload a good deal of the routine transcription drill that would otherwise need to be covered in class or handled with paper-and-pencil exercises using text as input rather than speech. The exercises were developed using Alvin, a software package that was developed for controlling stimulus presentation, screen layout, and the collection of responses for a wide variety of behavioral experiments. The original release of Alvin, which will be referred to as Alvin1, was described in Hillenbrand and Gayvert (2005). That software was later substantially revised, producing a version that is now known as Alvin2. Both Alvin1 and Alvin2 run only under Windows. A beta-test version of Alvin3 has separate executables available for Windows and Mac OS X. Alvin3 is quite different from Alvin2 internally but, with a few exceptions discussed later in this tutorial, those differences do not need to concern users. Control files (described below) that were developed for Alvin2 will run without modification under Alvin3, although at present Alvin3 does not support quite all of the functionality of Alvin2. The software can be downloaded from http://homepages.wmich.edu/~hillenbr/.

A menu is displayed when Alvin is started. Depending on what is set as the default menu, it may be necessary to change to the *Phonetics exercises* menu using the *Switch menu* button in the lower right. An exercise is chosen, and the user is prompted to enter a *subject identifier*. Any text string can be entered here as long as the spacebar is not used. Alvin tasks that do not appear in the menu can be run using the *Open experiment* (not *Open file*) option under the *File* menu, which displays a standard file-open dialog box. Tasks can be added to or deleted from the menu by editing the plain text file *menus.lua* in the *Lua* folder (e.g., 'c:\alvin2\lua\menus.lua').

It is important to note that the exercises were developed for a sophomore-level introductory phonetics course that is taken primarily by students pursuing either a major or a minor in speech pathology and audiology. For this reason, the exercises focus exclusively on English. As such, the exercises will not be well suited for more advanced coursework or for coursework in disciplines such as linguistics or second-language learning. However, only a modest time commitment would be required to adapt many of the exercises for coursework in other disciplines. For example, the

[a]Western Michigan University, Kalamazoo
[b]Gayvert Consulting, Fairport, NY
Correspondence to James M. Hillenbrand: james.hillenbrand@wmich.edu
Editor and Associate Editor: Jody Kreiman
Received June 6, 2014
Revision received August 18, 2014
Accepted October 23, 2014
DOI: 10.1044/2014_JSLHR-S-14-0149

Disclosure: *The authors have declared that no competing interests existed at the time of publication.*

consonant-transcription exercises could be easily adapted to incorporate Spanish sounds such as [β], [ɣ], and the trilled [r], and the collection of [ɑ]-[ɔ] exercises, described below, could be modified to focus on contrasts such as [u]-[y] in languages such as French.

Some instructors will be interested in using the exercises in their current form, and others may want to modify existing exercises or develop new ones of their own. Simply making the exercises available for students to use does not require any knowledge of how Alvin works. This tutorial will begin by addressing the more modest goal of describing the exercises that were developed. This will be followed by an explanation of the two main control files that govern Alvin's behavior: (a) the stimulus presentation file and (b) the script file.

Vowel Transcription Exercises

Figure 1 shows the screen layout for an exercise designed to introduce students to the phonetic symbols for 12 nominally monophthongal American English vowels. Before beginning the exercise, students can view written instructions by clicking the *Instructions* button at the bottom of the screen. On each trial, an /hVd/ syllable is presented to listeners, who are asked to click one of 12 buttons that are labeled with both a phonetic symbol and a key word (e.g., "heed," "hid," etc.). Listeners can repeat a stimulus by clicking the button labeled *Replay* before making a response. (There is a limit of two replays, although this default can be modified by changing a setting in the script file that controls the exercise. This and other aspects of the script file are discussed below.) Following the listener's response, the program provides feedback by briefly blinking the button corresponding to the correct response.

The /hVd/ syllables, which are presented in random order, are spoken by roughly equal numbers of men and women. The utterances are a subset of the recordings described in Hillenbrand, Getty, Clark, and Wheeler (1995). The issue of dialect is worth discussing. Most of the speakers were raised in southern Michigan. The speech patterns that typify this region represent a good example of the Northern Cities dialect described by Labov and colleagues (e.g., Labov, 1994; Labov, Yaeger, & Steiner, 1972). The qualities of some of these vowels will differ from those that are typical of other dialect regions. Two primary distinguishing features of these vowels are (a) /æ/ is raised and fronted and (b) /ɑ/ and /ɔ/ are lowered and fronted (Hillenbrand, 2003).[1] These features are quite conspicuous in some speakers. However, with the notable exception of difficulty distinguishing /ɑ/ and /ɔ/ (discussed below), we have not found that students raised outside of the Northern Cities dialect region have much difficulty identifying these vowels. Nevertheless, some instructors may prefer to use a set of recordings that are a better match to the dialect spoken in their part of the country. In terms of the structure of the software, substituting a different set of signals in place of the Michigan recordings is a simple matter.

When the exercise is completed, students are given the option of repeating any trials in which incorrect responses were given. A text file is then displayed that shows the student's performance. This file shows two confusion matrices with the vowel intended by the talker going down the rows and listener responses going across the columns.[2] One matrix shows responses in numerical form, and a second shows the same data as percentages. Also shown are percent correct figures for each of the 12 vowels along with the listener's overall percent correct for all vowels. Students should be told that they should not expect to score 100% on this exercise. Formal listening tests have shown that the intelligibility of these utterances for phonetically trained listeners averages about 95% (Hillenbrand et al., 1995; see also a very similar figure from Peterson & Barney, 1952). A percent correct score in the low 90s and above generally indicates that the student has learned the sound-to-symbol associations, which is the goal of the exercise. Students will generally be interested mainly in the overall percent correct figure, but percent correct figures for individual vowels and inspection of confusion matrices can sometimes provide useful information. For example, it is not uncommon for students to confuse /ɑ/ and /ɔ/. Students from dialect regions that do not distinguish these vowels have particular difficulty with /ɑ/ and /ɔ/, but we also find this for an important minority of students who are raised in southern Michigan, a region in which the /ɑ/-/ɔ/ distinction is typically quite well maintained. A fairly small minority of students also have difficulty learning sound–symbol associations for a cluster of back vowels including /u/, /ʊ/, and /ʌ/. Problems such as these are quite easy to see in the confusion matrices.

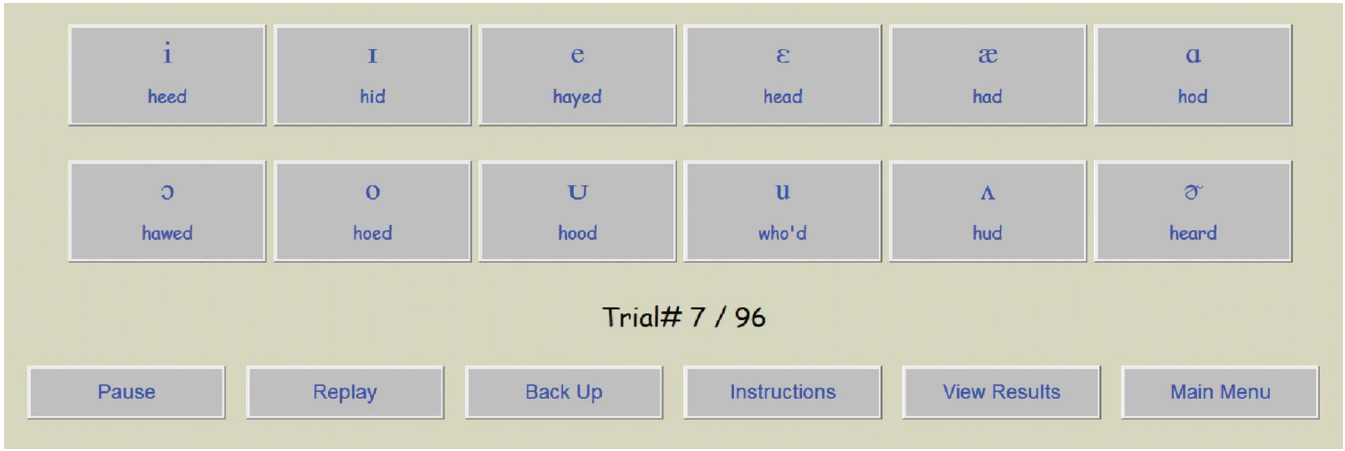
A second vowel transcription exercise is included that is identical to the one described above except that each button is labeled using only the phonetic symbol rather than the symbol and a key word. Students who do not feel that they need the key words might want to skip directly to the second exercise.

Problems with /ɑ/ and /ɔ/ are common enough that four optional exercises were developed that focus entirely on this pair. In the first of these exercises, a pair of /hVd/ syllables is presented. On a random half of the trials, /hɑd/ is presented first, followed by /hɔd/, and on the other half of the trials, the order is reversed. The listener's task is to choose between buttons labeled */ɔ/ First* and */ɔ/ Second*. The second /ɑ/-/ɔ/ exercise is identical except that word pairs such as *cot–caught* and *odd–awed* are used instead of

[1]The symbols that are used for the vowel exercises should be thought of as broad phonemic rather than narrow phonetic symbols. In terms of detailed phonetic quality, most of the tokens of /ɑ/ and /ɔ/ in the Hillenbrand et al. (1995) database are more accurately represented using [a] and [ɒ], respectively (MacKay, 1987).

[2]Because only plain text files are displayed by Alvin, codes are used to represent the vowel categories. The text codes for the vowels are /i/: iy, /ɪ/: ih, /e/: ei, /ɛ/: eh, /æ/: ae, /ɑ/: ah, /ɔ/: aw, /o/: oa, /ʊ/: oo, /u/: uw, /ʌ/: uh:, and /ɚ/: er.

Figure 1. Screen layout for the first of two vowel transcription exercises.

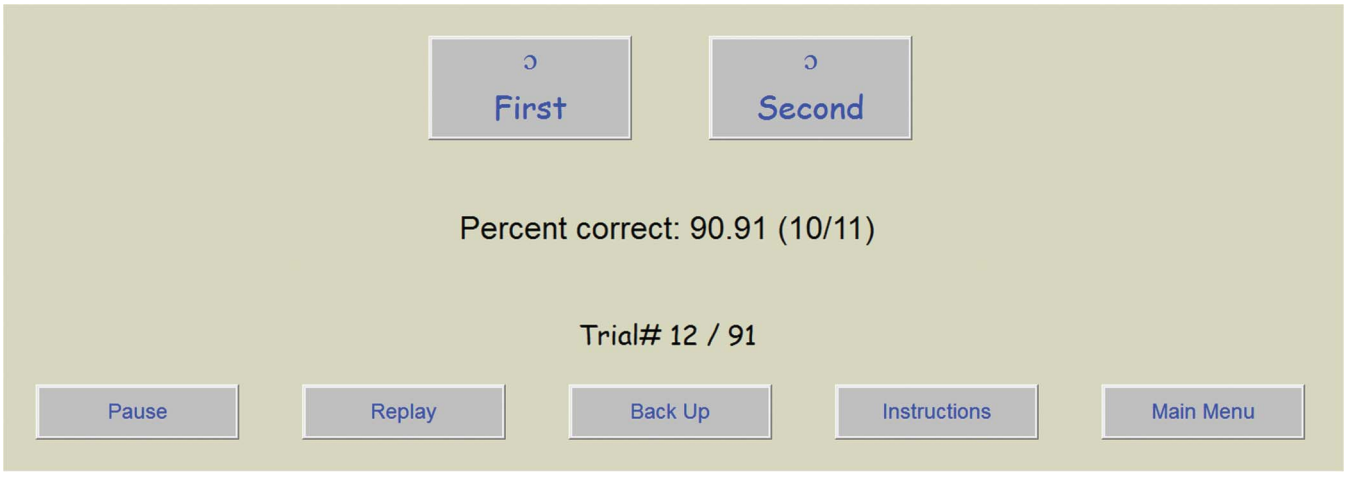


/hVd/ syllables. A results file is not displayed at the end of these two exercises. Instead, a running percent correct figure is displayed on the screen and updated with each trial (see Figure 2).

A second pair of /ɑ/-/ɔ/ exercises is structured differently. On most trials, listeners hear a word that includes either /ɑ/ or /ɔ/, but on a minority of trials (~17%), the word will not include either of these vowels. The listener's task is to choose among three buttons labeled */ɑ/*, */ɔ/*, and *neither*. Feedback is provided in the first of these exercises; the second exercise is identical but without feedback. Results similar to those described earlier are displayed when the exercise is completed.

A final point about the /ɑ/-/ɔ/ exercises is worth discussing. Although many of our students have little difficulty with this vowel pair, others will have to put a fair amount of effort into learning to identify these vowels consistently. Instructors who are primarily involved in training students headed for careers in speech pathology or audiology may well feel that the time some students will need to invest in this effort may not be well spent given that the direct clinical significance of distinguishing these two vowels is, at best, limited. This is a defensible view, and instructors may well want to simply skip these exercises. However, we continue to encourage students to make their best effort to learn to hear this distinction because the development of a discriminating ear for speech is of considerable importance, for both clinicians and students who are primarily interested in research problems in phonology and experimental phonetics. These kinds of listening skills appear to be quite specific to speech, and they do not develop over a short period of time (e.g., Hillenbrand, Canter, & Smith, 1990). It is our view that extended practice in learning to hear speech sound distinctions that do not come easily to listeners can play some role in the development of these kinds of skills.

Figure 2. Screen layout for the second of four exercises focusing on /ɑ/ and /ɔ/.



Diphthong Transcription Exercises

Separate exercises were developed for nonrhotic diphthongs (see Figure 3a) and rhotic diphthongs (see Figure 3b). Little explanation is needed because the structure of these tasks is identical to the vowel exercises described above. Transcription conventions for diphthongs vary quite a bit across phoneticians. These exercises use the transcription conventions in MacKay (1987), the textbook that is used by our students. Relabeling the buttons to use a different set of phonetic symbols is fairly simple. This will be discussed below. Also, note that a distinction is made between [ai] and [ʌi] (sometimes transcribed as [əi]); for example, *hide* ([haid]) versus *height* ([hʌit]) or *eyes* ([aiz]) versus *ice* ([ʌis]). This distinction, typically [ʌi] preceding unvoiced consonants and [ai] in other contexts, is maintained in most dialects of English. Some phonetics instructors, however, do not distinguish between these allophones. The exercise can be modified to collapse [ai] and [ʌi] into a single category.

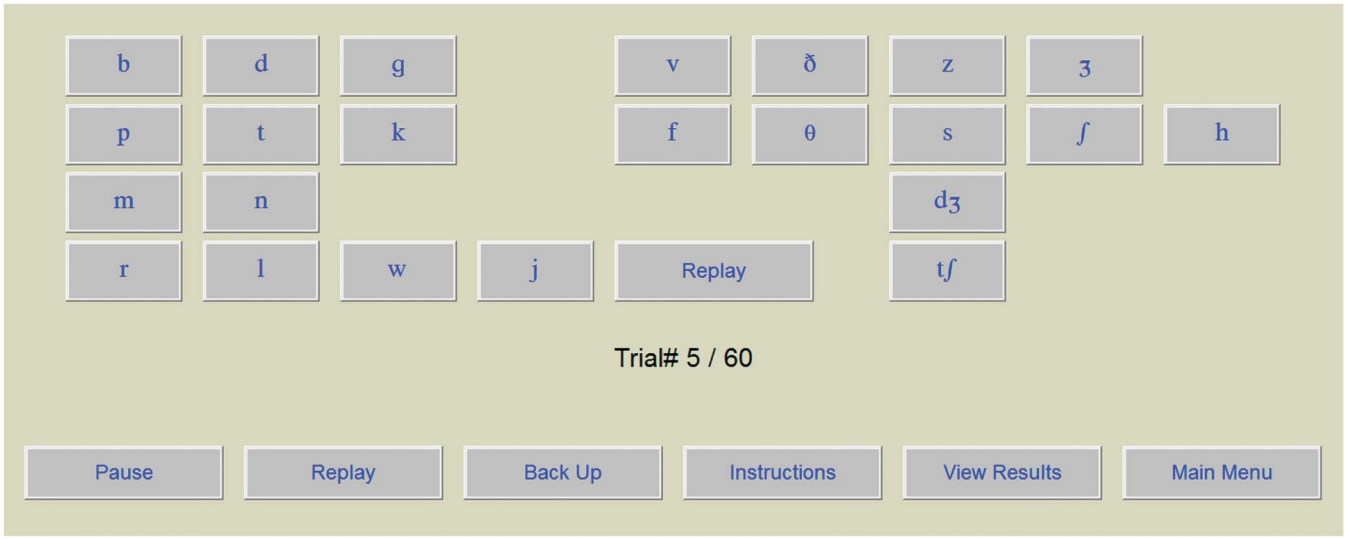
It has been our experience that most students readily appreciate that the diphthongs in words such as *lied* and *light* are pronounced differently. However, for reasons that are not entirely clear, a fair number of students have difficulty consistently identifying these sounds despite the fact that nearly all of our students distinguish these sounds in their own speech. Further, quite a few students appear to derive little or no benefit from being told about the (less than 100% consistent) rule about the voicing property of the sound that follows the diphthong. As with the /ɑ/-/ɔ/ distinction, our view is that encouraging students to put some effort into learning to distinguish these two sounds can be justified not for the direct clinical relevance of this specific ability but rather as a way to encourage the development of a more discriminating ear for speech.

Consonant Transcription

Figure 4 shows the screen layout for the second of two exercises that provide practice in using 23 consonant symbols. The structure of this exercise is identical to the vowel transcription exercises described above. As with the

Figure 3. Exercises for nonrhotic (a) and rhotic diphthongs (b).

Figure 4. Screen display for the second of two consonant transcription exercises.



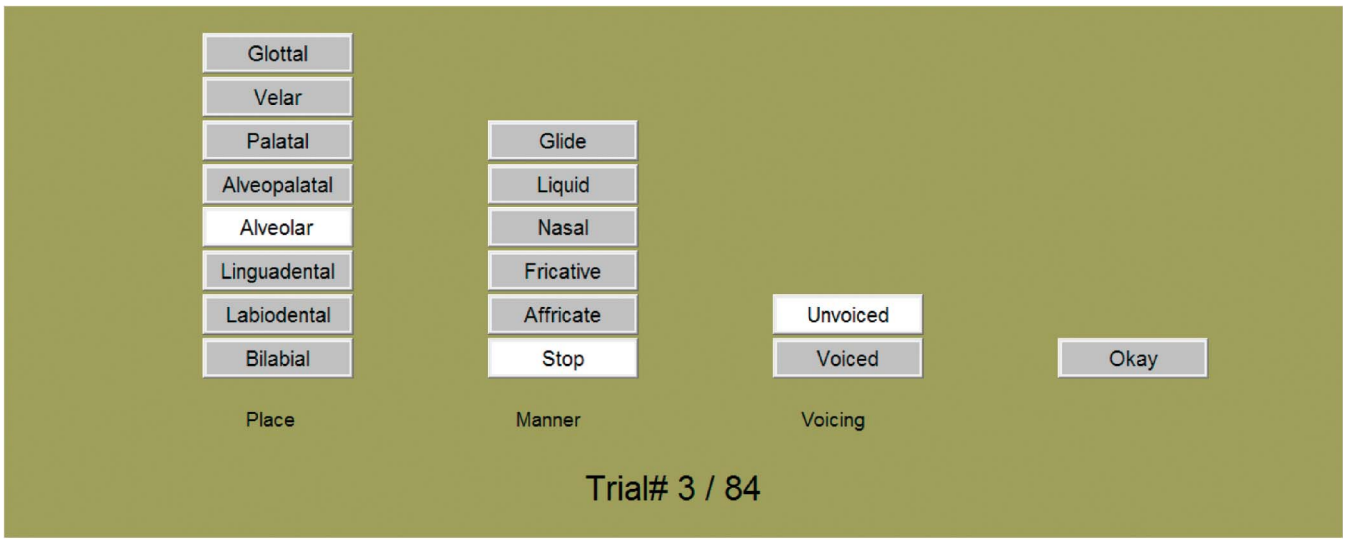
vowels, the first exercise uses buttons that are labeled with both phonetic symbols and key words, and the second exercise uses only phonetic symbols. A text file of the same form as the vowel exercise (confusion matrices, percent correct for each symbol, and overall percent correct) is displayed when the exercise is finished. The test signals are a subset of the consonant–vowel syllables described by Shannon, Jensvold, Padilla, Robert, and Wang (1999).

Although it is not common, students occasionally have difficulty distinguishing /θ/ and /ð/, so an optional exercise focusing on this pair was developed. The exercise appears in the menu as *Theta vs. bar-d*. On each trial, students hear a word containing either /θ/ or /ð/ in various positions in the words. The listener's task is to choose between buttons labeled with the symbols /θ/ and /ð/. Feedback and a report detailing the student's performance are provided in the same manner as in previous exercises.

Place, Manner, and Voicing

Figure 5 shows the response alternatives for the place, manner, and voicing exercise. A consonant–vowel syllable is presented on each trial, and listeners are asked to select one button from each of the three columns to specify the

Figure 5. Response alternatives for the place, manner, and voicing exercise. The student's choices are shown in a white background. Following the trial, the correct feature descriptors are shown with a blue background.



place, manner, and voicing of the consonant that was presented. The listener clicks the *Okay* button to end the trial, and feedback is provided by highlighting buttons corresponding to the three correct features. When the exercise finishes, three files are displayed showing performance separately for place, manner, and voicing.[3]

Word Transcription

Several word-transcription exercises have been developed. On each trial, listeners hear a word or short phrase, and their task is to enter a phonetic transcription of the utterance in a text box. After entering a transcription and clicking the *Next* button to end the trial, the correct transcription is displayed on the screen (see Figure 6). If the transcription entered by the student is correct, this transcription is displayed in blue; otherwise, it is displayed in red. Cumulative percent correct for the exercise is also displayed on the screen and updated with each trial. The keyboard is remapped to allow phonetic symbols to be entered. Appendix A shows the keyboard mapping. With a few exceptions, the keys bear some reasonable relationship to the phonetic symbols they represent (e.g., 'U' = /ʊ/, 'N' = /ŋ/, 'S' = /ʃ/, etc.), so students typically learn the mapping quickly. For some words, two or more alternate transcriptions will be accepted as correct; for example, the second vowel in a word such as *purpose* might be transcribed using either /ə/ or /ɪ/. If an instructor prefers to accept just one transcription as correct, a simple change can be made to the stimulus presentation file, which is described in more detail under the section "How Alvin Works." A single line from the stimulus presentation file for Word Transcription Exercise 4 ('wordtranscription04.stim') is shown below:

purpose purposemod.wav "pRp&s or pRpIs"

The two alternate transcriptions, which must be enclosed in double quotes, are separated by the word *or*. To make just one of these transcriptions acceptable, simply delete the *or* and one of the transcriptions. In a similar manner, other alternative transcriptions can be added. Some of the word-transcription exercises were designed to focus primarily on a specific set of sounds (e.g., vowel symbols in simple words, schwa, diphthongs, etc.), and others, identified in the menu as *General practice,* sample a variety of sounds.

Reverse Transcription

Reverse transcription exercises present students with a phonetic transcription, and their task is to enter the word into a text-entry box in ordinary orthography. The correct word is then displayed on the screen. As with the word transcription exercises, the text is displayed in blue if the correct word was entered and in red if the word was incorrect (see Figure 7).

How Alvin Works

As discussed above, instructors who are interested in running the exercises in their current form do not need to understand how Alvin works, and this section can be skipped. However, developing new exercises or making changes to existing ones requires some understanding of how the program is organized.

Alvin is controlled by two plain-text files: (a) a stimulus presentation file with the default extension '.stim' and (b) a script file with the default extension '.luax.' Script files are written in Lua (http://www.lua.org/) although modifying existing exercises or creating new exercises that are similar to the ones that are provided does not require any real understanding of Lua. The stimulus presentation file includes a list of the stimuli that are to be presented, which is the only required element in these files. However, these files usually contain other optional information that might be useful in analyzing the listener's performance. A few lines of the '.stim' file for a vowel transcription exercise are shown below:

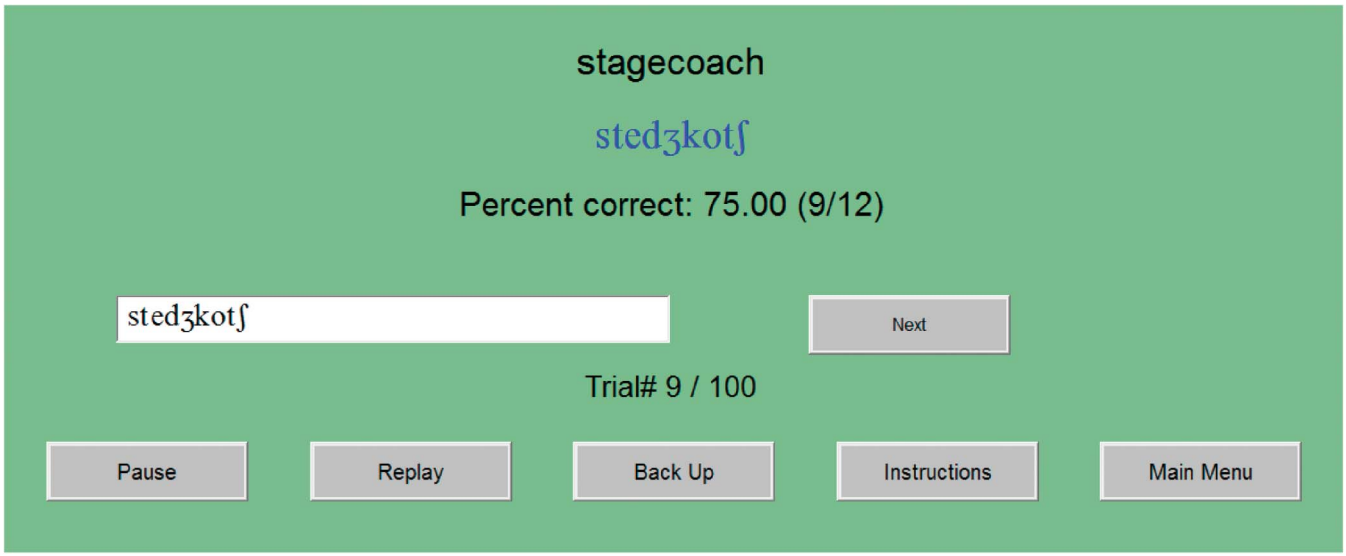
ae	m	m01aemod.wav
ah	m	m01ahmod.wav
aw	m	m01awmod.wav
eh	m	m01ehmod.wav
ei	m	m01eimod.wav

In the first column are codes for the vowel that was spoken (ae = /æ/, ah = /ɑ/, aw = /ɔ/, etc.), the second column holds a code for talker sex (man or woman), and the third column holds the name of the sound file. The optional information in the first two columns can be written to the output file created by Alvin and later used to analyze the results (see the 'formatResultLine' function described in Appendix B). Unlike the original version of Alvin, there is no fixed format for the '.stim' file, although the order of information in this file needs to match certain key information in the '.luax' file. Alvin copies the text strings that appear in the '.stim' file into a structure called 'exp.stimulus.' A straightforward coding scheme is used: The text string in the first column of the '.stim' file (*ae* for the first line in the example above) is copied into 'exp.stimulus[1],' the string in the second column (*m*) is copied into 'exp.stimulus[2],' and the string in the third column (*m01aemod.wav*) is copied into 'exp.stimulus[3].' Everything in the 'exp.stimulus' structure is stored as a character string. The character strings in 'exp.stimulus' are then used in a variety of ways to control functions such as presenting stimuli, providing feedback, writing data to a results file, and checking whether listeners' responses were correct or incorrect.

The best way to understand how this works is to study the script file in Appendix B, which is a heavily annotated version of the '.luax' file for the first vowel transcription

[3]Text codes are used to represent the place, manner, and voicing features in the confusion matrices. The place codes are al = alveolar, ap = alveopalatal, bd = labiodental, bl = bilabial, gd = linguadental, gl = glottal, pa = palatal, and ve = velar. The manner codes are st = stop, af = affricate, fr = fricative, na = nasal, li = liquid, and gl = glide. The voicing codes are v = voiced and u = unvoiced.

Figure 6. Screen layout (at the end of a trial) in a word-transcription exercise. The correct transcription is shown beneath the test word at the end of the trial; it will appear in red if the transcription is incorrect.



exercise. The code that follows is the function that controls the presentation of the audio signals in the vowel transcription exercise. The function is called 'exp.presentation' and it is a required element of all '.luax' files. This example is intended to show how information in the '.stim' file is hooked together with information in the '.luax' file:

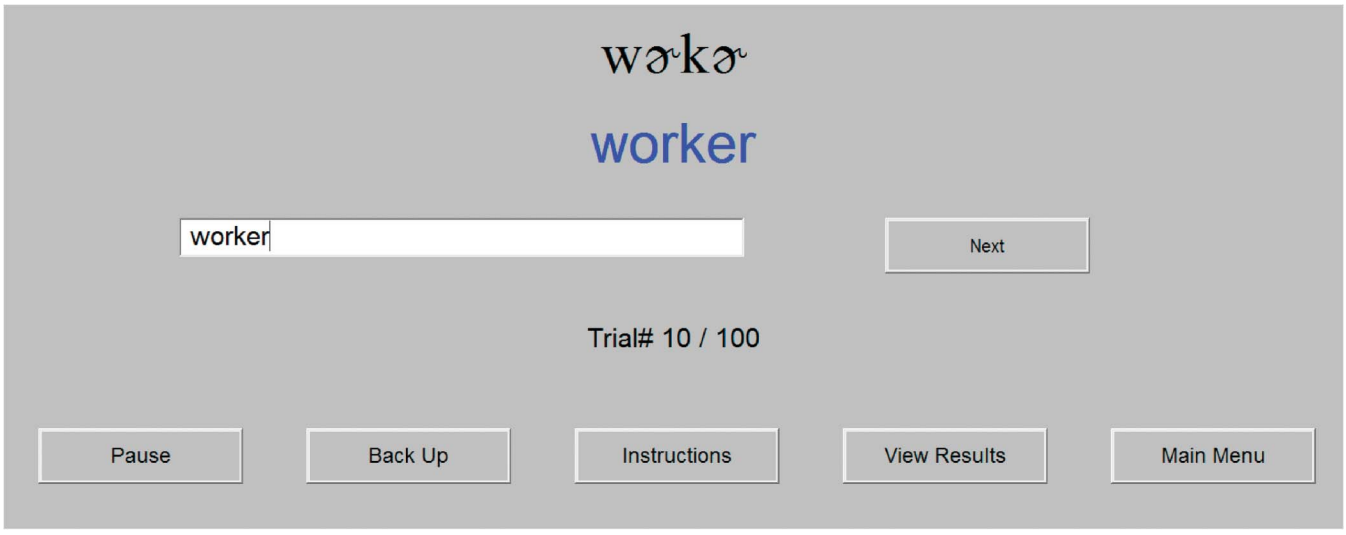
```
function exp.presentation()
   exp.playAudioFile(exp.stimulus[3])
end
```

This function simply instructs Alvin to play the audio signal whose name appears in Column 3 of the '.stim' file. The name of the sound file can appear in any column of the '.stim' file, but the call to 'exp.playAudioFile' needs to match; for example, if the name of the sound file were to appear in Column 2 of the '.stim' file, the play function would look like this:

```
exp.playAudioFile(exp.stimulus[2])
```

The script file in Appendix B provides quite a few examples of text strings in the 'exp.stimulus' structure being used to control a variety of functions related to presenting stimuli, storing responses, providing feedback, evaluating responses as correct or incorrect, and specifying the information that is written to the '.res' output file.

Figure 7. Screen layout (at the end of a trial) in a reverse-transcription exercise. The correct word is shown beneath the transcription at the end of the trial; it will appear in red if the word is incorrect.



A second example from a more complicated task (not included in the phonetics exercises) might help to illustrate the relationship between '.stim' and '.luax' files. This experiment tests ABX discrimination for pairs of signals drawn from a nine-step synthetic continuum varying from /we/ to /re/. Three stimuli are presented on each trial. The first two signals (A and B) differ from one another by two steps on the nine-step continuum, and the third stimulus (X) is an exact match to one of the first two signals. The listener's task is to choose between buttons labeled *1* and *2* to indicate whether the third stimulus matches stimulus 1 or stimulus 2. Shown below are a few lines from the '.stim' file:

```
1 3 1 1 wr01.wav 500 wr03.wav 700 wr01.wav
1 3 3 2 wr01.wav 500 wr03.wav 800 wr03.wav
2 4 2 1 wr02.wav 500 wr04.wav 900 wr02.wav
2 4 4 2 wr02.wav 500 wr04.wav 500 wr04.wav
3 5 3 1 wr03.wav 500 wr05.wav 500 wr03.wav
```

The first three values on each line give the stimulus numbers (1 through 9) of the three stimuli that will be presented on the trial, and the fourth value gives the correct response; that is, whether the third stimulus matches Stimulus 1 or Stimulus 2. The names of the three sound files are given in Columns 5, 7, and 9. The values in Columns 6 and 8 give the interstimulus intervals in milliseconds (this task was created to show how interstimulus intervals can be varied from one trial to the next). The stimulus presentation function for this task is shown below:

```
function exp.presentation()
   exp.playAudioFile(exp.stimulus[5])
   exp.isi(exp.stimulus[6])
   exp.playAudioFile(exp.stimulus[7])
   exp.isi(exp.stimulus[8])
   exp.playAudioFile(exp.stimulus[9])
end
```

For the first line of the '.stim' file (1 3 1 1 wr01.wav 500 wr03.wav 700 wr01.wav), Alvin will (a) play 'wr01.wav,' wait 500 ms, (b) play 'wr03.wav,' wait 700 ms, then (c) play 'wr01.wav.' The full '.stim' and '.luax' files can be found in the 'wrabxnofeedback' directory, which is one level down from the 'sampleexperiments' directory. If the full Alvin installation is run, a large number of other examples of '.stim' and '.luax' files can be found, which illustrate a variety of other features of Alvin, including response modalities other than the buttons and text boxes that are used in the phonetics exercises. A final comment to those who may be new to programming is that Lua, like all other programming languages, is quite fussy about apparently minor details such as quotation marks, commas, and distinctions such as the difference between square brackets and curly braces. For example, in Appendix B, a line in the 'defineExperiment' function reads:

```
controlFile="vowels.stim",
```

If the quotation marks (which tell Lua to treat 'vowels.stim' as a character string) are omitted, Alvin will not find the '.stim' file, and the task will not run. The comma at the end of the line is also needed. Further, Lua does not have any way to recognize either of these mistakes as errors, so it is up to the programmer to figure out why the task does not run, which can sometimes be quite time consuming. Close attention to details such as these can avoid a good deal of grief.

Variations in Transcription Conventions

Transcription practices are far from uniform across instructors. As noted earlier, the exercises follow the transcription conventions used in MacKay's (1987) textbook, but these practices may well be at odds with the ones that are taught by other instructors. There is particularly large variation in the symbols that are used to transcribe diphthongs. For example, the diphthong in a word such as *buy* is variously transcribed as /ɑi/, /ɑɪ/, /ai/, or /aɪ/. The Alvin transcription exercises use /ai/. In a similar manner, the rhotic diphthong in *beer* might be transcribed as /iɚ/, /ɪɚ/, /ir/, or /ɪr/. This variation in transcription habits occurs mainly because the vowel qualities that comprise diphthongs typically do not quite correspond to any monophthongal vowel. For example, the first vowel in *bear* is intermediate between /e/ and /ɛ/ (Clark & Hillenbrand, 2003). In a similar manner, the first vowel in *buy* is intermediate between /ɑ/ and /a/, and the second vowel is intermediate between /i/ and /ɪ/. Whatever the explanation, modifying the exercises to conform to a different set of transcription conventions is not difficult. The line below is from the '.luax' file for the first rhotic diphthong exercise:

```
button.new{name="ir", x=15, label="\\69D4",
value="ir", label2="beer"}
```

This line of code creates a button that is labeled with the International Phonetic Alphabet symbols for *iɚ* using the *SILDoulosIPA* phonetic symbol font. (Fonts are defined in the plain text file 'fonts.lua' in the 'Lua' directory.) This is done with *label = "\\69D4"*, where 69 is the hexadecimal code for lowercase *i*, and D4 is the code for schwar. To change this label to *ɪɚ*, the 69 needs to be changed to 49, the code for /ɪ/; that is, *label = "\\69D4"* would become *label = "\\49D4"* (note that the double backslashes and quotation marks are required). Similar changes can be made to other exercises so that students do not have to contend with learning one set of transcription practices in lecture and/or their textbook while adopting another set for use in the exercises. Appendix C explains the methods that can be used to find the hexadecimal codes for phonetic symbols and other special characters, along with much simpler methods for specifying these characters using Alvin3.

Changes can also be made to the '.stim' files used in the word transcription exercises. For example, assume that an instructor prefers that students transcribe words such as *hide* as /haɪd/ rather than /haid/, the transcription that is preferred by MacKay (1987). The third column of the '.stim' files that are used with the word transcription exercises

(e.g., 'wordtranscription01.stim' in 'c:\alvin2\phonetics\wordtranscription') holds the correct transcription for the word that was presented. All that is needed for this example is to change this text string from *haid* to *haId*. Further, if an instructor prefers to accept either of these transcriptions as correct, the text string in Column 3 can be changed to "haid or haId." Note that, with any entry containing internal spaces, the quotation marks are needed so that Alvin will treat this entry of "haid or haId" as a single text string rather than three separate strings.

Alvin Output Files

An output file with the extension '.res' is created for every exercise. This file holds the listener's responses along with other information that might be needed to analyze the student's performance. A few lines from a vowel exercise are shown below:

1	oo	m	m04oomod.wav	uw
2	ei	w	w03eimod.wav	ei
3	oo	m	m01oomod.wav	oo
4	uh	m	m04uhmod.wav	uh
5	uh	w	w02uhmod.wav	ah
6	oa	m	m03oamod.wav	oa
7	ei	m	m03eimod.wav	ei

Going across the columns, the file shows the trial number, a code for the vowel intended by the talker, a code for speaker sex ('m' or 'w'), the name of the sound file, and a code for the listener's response, using the same scheme for coding vowel identity as the data in Column 2. For many but not all of the exercises, the Alvin script runs a separate, stand-alone executable that analyzes the '.res' files. For example, the vowel and consonant transcription exercises are analyzed using a program called 'cmat.exe,' which generates confusion matrices ('.cm' files) and calculates percent correct for each of the symbols along with overall percent correct. (In Alvin3, all of the calculations are made directly in the '.luax' file, and no separate executables are used.) For other exercises, performance is calculated and displayed on the screen as the task progresses. For these exercises, the raw '.res' file is the only output file that is created. Appendix D lists the names and locations of the output files that are created for each of the exercises.

What Is Missing From the Exercises?

Although the collection of phonetics exercises is relatively large, it is not complete. There are a few areas in which exercises would be useful, but they have not yet been developed. In our experience, students tend to have particular difficulty transcribing words with allophones such as flaps, glottal stops, and syllabic consonants. Developing word transcription exercises focusing on these sounds would be straightforward, and we expect to have at least one exercise devoted to these sounds running fairly soon.

Another area of need is an exercise providing students with practice in identifying stress patterns. In lecture, transcription conventions for marking primary stress, secondary stress, and unstressed syllables are discussed, and some group practice is provided in class. However, in terms of performance expectations, we ask that students learn to identify primary stress only. Most of our students have little or no difficulty with this, but we have consistently found a minority of perhaps 10%–15% who have quite a bit of trouble learning to identify the strongest syllable in a word. An exercise is needed that provides practice in marking stress patterns.

There are undoubtedly other aspects of phonetics education in which additional exercises would be useful. One motivation for writing this description was the hope that instructors might develop exercises of their own that could be shared with others. An instructor who develops an exercise can contact the author, and the files can be added to the installation script and made available to other users.

Acknowledgments

Development of the Alvin software was supported in part by a grant from the National Institutes of Health (R01-DC01661) to Western Michigan University. Thanks to Claire Carpenter for helpful comments on an earlier draft.

References

Clark, M. J., & Hillenbrand, J. M. (2003). Quality of front vowels before /r/. *Journal of the International Phonetic Association, 33,* 1–16.

Hillenbrand, J. M. (2003). American English: Southern Michigan. *Journal of the International Phonetic Association, 33,* 121–126.

Hillenbrand, J. M., Canter, G. J., & Smith, B. L. (1990). Perception of intraphonemic differences by phoneticians, musicians, and inexperienced listeners. *The Journal of the Acoustical Society of America, 88,* 655–662.

Hillenbrand, J. M., & Gayvert, R. A. (2005). Open source software for experiment design and control. *Journal of Speech, Language, and Hearing Research, 48,* 45–60.

Hillenbrand, J. M., Getty, L. A., Clark, M. J., & Wheeler, K. (1995). Acoustic characteristics of American English vowels. *The Journal of the Acoustical Society of America, 97,* 3099–3111.

Labov, W. (1994). *Principles of linguistic change*. Cambridge, MA: Blackwell.

Labov, W., Yaeger, M., & Steiner, R. (1972). A quantitative study of sound change in progress. *Report on National Science Foundation Contract NSF-FS-3287*.

MacKay, I. (1987). *Phonetics: The science of speech production* (2nd ed.). Boston, MA: Little Brown.

Peterson, G., & Barney, H. L. (1952). Control methods used in a study of the vowels. *The Journal of the Acoustical Society of America, 24,* 175–184.

Shannon, R. V., Jensvold, A., Padilla, M., Robert, M., & Wang, X. (1999). Consonant recordings for speech testing. *The Journal of the Acoustical Society of America, 106,* 71–74.

Appendix A

Keyboard mapping for the phonetic symbols that are used in the Alvin word transcription exercises. Symbols that are transparent (b, d, g, etc.) are not included.

Vowels		
/i/	*heed*	i
/ɪ/	*hid*	ɪ
/e/	*hayed, bait*	e
/ɛ/	*head*	E
/æ/	*had*	@ (mnemonic: this is the *at* (/æ/) symbol)
/ɑ/	*hod, pod*	A
/ɔ/	*hawed, caught*	c (mnemonic: open-o is a backwards 'c')
/o/	*hoed, boat*	o
/ʊ/	*hood*	U
/u/	*who'd, boot*	u
/ʌ/	*hud, but*	^ (<SHIFT> '6')
/ə˞/	*heard*	R
/ə/	***a**bout, mantr**a***	& (no mnemonic: this one has to be memorized)
Consonants		
/θ/	***th**in*	T
/ð/	***th**en*	D
/ʃ/	***sh**oe*	S
/ʒ/	*meas**u**re*	Z
/ʔ/	*uh-oh, bu**tt**on*	**/** (mnemonic: '/' is just below '?' key)
/tʃ/	***ch**ur**ch***	tS
/dʒ/	***j**ud**g**e*	dZ
/ʍ/	***wh**ich* / ***wh**ether*	M (mnemonic: /ʍ/ looks a little like 'M')
/ŋ/	*si**ng***	N
/ɾ/	*bu**tt**er*	F (mnemonic: F = flap)
Diacritics		
̩	*syllabic* (e.g., [n̩])	` (back quote; upper left of keyboard)
̃	*nasalized* (e.g., [õ])	) (right parenthesis)
ʰ	*aspiration* (e.g., [t^{h}])	H (upper case h)

Appendix B (p. 1 of 3)

Annotated script file for Vowel Transcription Exercise 1 ('c:\alvin2\phonetics\vowels\vowelsymbols1.luax'). Comments are shown in italics.

Experiment: *Vowel identification*
Presentation: *A single vowel*
Response: *Button click indicating which vowel was heard*
Feedback: *Button blink of the button corresponding to the correct response*
Stimulus: *<vowel> <m | w> <filename>*
Output: *<trial index> <index> <filename> <response> <reaction time> <replay count>*

The function below sets several options that control how the task is run. Many more options are available than the ones used below.

```
defineExperiment{
```

stimulusDir="signals",	*This gives the name of the subdirectory in which sound files are found; for example, if the '.luax' and '.stim' files are in 'c:\alvin2\phonetics\vowels,' Alvin will look for the sound files in a directory called 'signals' that is one level below 'vowels.'*
controlFile="vowels.stim",	*This gives the name of the stimulus presentation file.*
shuffle=1,	*Stimuli will be presented once in random order. With 'shuffle=2,' the full set of signals will be presented in random order, the list will be reshuffled, then presented a second time. If 'shuffle=0' is used, signals will be presented in the order in which they are listed in the '.stim' file.*
helpFile="instructions1.txt",	*This gives the name of the instructions file, which is displayed when the* Instructions *button is clicked.*
replayLimit=2,	*An optional* Replay *button allows listeners to repeat a stimulus. This command sets a limit of two replays.*
resultFileSuffix="vsym1",	*Listeners enter a 'subject identifier' when an exercise is started, for example, the student's initials. With this command, the character string "vsym1" will be appended to the subject identifier when the file holding the student's (unanalyzed) results is created. In this example, with the subject identifier "jmh," Alvin will create the results file 'jmhvsym1.res.'*
allowRepeatingIncorrectResponses=true	*With this option set, listeners will be given the opportunity of repeating trials with incorrect responses.*

```
}
```

The function below controls the presentation of stimuli. This particular presentation function is quite simple: It plays the sound file in Column 3 of the stimulus presentation ('.stim') file, referenced below as 'exp.stimulus[3].' Other examples can be found in the 'sampleexperiments' directory; for example, 'wrabx.luax' shows the 'exp.presentation' function for an ABX discrimination task.

```
function exp.presentation()
   exp.playAudioFile(exp.stimulus[3])
end
```

The function below blinks the correct button. The buttons are named according to the vowel, so the string in the first column of the stimulus file ('exp.stimulus[1]') is sent to 'button.blink().'

```
function exp.showFeedback()
   local vowel = exp.stimulus[1]
   button.blink(vowel)
end
```

The function below controls what information is written to the '.res' output file (the raw, unanalyzed data). It also controls how that information is formatted. In this example, the data to be written to the '.res' file consist of the trial number ('exp.trialIndex'), the strings in Columns 1–3 of the '.stim' file ('exp.stimulus[1],' . . .), the listener's response ('exp.response'), the response latency ('exp.reactionTime'), and the number of replays for the trial ('exp.replayCount'). The formatting information ("%4d %2s %-12s", etc.) follows the conventions used in C for functions such as 'printf().' These formatting conventions are much simpler than they may seem to someone who is seeing them for the first time. For example, the "%4d" in the format statement below specifies the printing of an integer (indicated with the 'd') using a field width of 4; the "%-12s" specifies the printing of a text string with a field width of 12. The minus sign indicates that the text should be left justified, the default being right justification. Numerous explanations can be found on the web. Search for something like "C format specifiers." Read the page that most resembles English and that provides a decent collection of examples. The 'printf' page on http://www.codingunit.com looks fairly straightforward.

```
function exp.formatResultLine()
   return string.format("%4d %2s %-12s %2s %s %4d %2d",
      exp.trialIndex, exp.stimulus[1], exp.stimulus[2], exp.stimulus[3],
      exp.response, exp.reactionTime, exp.replayCount)
end
```

Appendix B (p. 2 of 3)

Annotated script file for Vowel Transcription Exercise 1 ('c:\alvin2\phonetics\vowels\vowelsymbols1.luax'). Comments are shown in italics.

The 'layout' function below controls the screen display and allows the creation of buttons, text-entry boxes, sliders, and other gadgets that can be used to elicit responses from listeners.

```
function exp.layout()
```

The command below sets the screen background color. The script 'colorscheme.luax' (in 'c:\alvin2\demos\colorscheme') displays the colors that are available.

```
canvas.backcolor("red")
```

The command below sets several defaults for the buttons that will be displayed on the screen. This function simply saves you the trouble of repeating these arguments when each button is defined. The parameters 'h' and 'w' set the default height and width of the buttons. The numerical arguments are expressed in percentages of screen height and width, not in pixels. The 'y' parameter sets the default y location of the buttons that follow as a percentage of screen height. The 'command' argument sets the default function that is called when the button is clicked; 'font' and 'font2' control the default fonts that are used for 'label' and 'label2' (see below). The 'fontsize' argument sets the default font size for 'label.' Once again, the numerical value is a percentage rather than an absolute font size. Finally, the default text color is set to blue.

```
button.setdefaults{w=14, h=14, y=80, command=exp.recordButtonResponse, font="Phonetics",
   fontsize=3.5, font2="Comic", textcolor="blue"}
```

The 'button.new' command defines a new button and allows the characteristics of the button to be defined. Each of the 12 buttons is given a name consisting of a two-character text string that identifies the vowel ("iy," "ih," etc.). The 'x' location of the button is then set, followed by settings for 'label' (upper) and 'label2' (lower)—text strings that are displayed on each button (see Figure 1). Numeric codes (e.g., "\\69") are needed to display symbols that do not correspond to one of the keys on a standard keyboard. These codes can be found in the lower left of the Windows Character Map utility (see Appendix C). However, in order to use Character Map to find these numeric codes, the font that is being used needs to be installed. See 'Phonetic font installer' toward the bottom of http://homepages.wmich.edu/~hillenbr/204.html. The value argument sets the text string that is stored each time the button is clicked.

```
button.new{name="iy",     x=5,     label="\\69",    label2="heed",     value="iy"}
button.new{name="ih",     x=20,    label="\\49",    label2="hid",      value="ih"}
button.new{name="ei",     x=35,    label="\\65",    label2="hayed",    value="ei"}
button.new{name="eh",     x=50,    label="\\89",    label2="head",     value="eh"}
button.new{name="ae",     x=65,    label="\\41",    label2="had",      value="ae"}
button.new{name="ah",     x=80,    label="\\E5",    label2="hod",      value="ah"}
```

Change the default 'y' position of the buttons to 60 (60% of screen height), positioning the second row of buttons a little below the first row.

```
button.setdefaults{y=60}

button.new{name="aw",     x=5,     label="\\D8",    label2="hawed",    value="aw"}
button.new{name="oa",     x=20,    label="\\6F",    label2="hoed",     value="oa"}
button.new{name="oo",     x=35,    label="\\55",    label2="hood",     value="oo"}
button.new{name="uw",     x=50,    label="\\75",    label2="who'd",    value="uw"}
button.new{name="uh",     x=65,    label="\\FA",    label2="hud",      value="uh"}
button.new{name="er",     x=80,    label="\\FB",    label2="heard",    value="er"}
```

With the command below, Alvin will display the trial number and total number of trials on the screen as the task progresses.

```
exp.addTrialLabel{label="Trial#", y=40, w=100, align="center", font="Comic"}
```

The command below defines a collection of commonly used buttons that appear toward the bottom of the screen (see Figure 1). If any of the buttons are not wanted, those entries can simply be removed from the list of arguments. Note that the syntax for the Results button is different. This allows the Results button to be enabled when the task is started.

```
exp.addStandardButtons{buttons={"start", "replay", "backup", "help", {name="results",
   enabled=true}, "main"}}

end
```

Appendix B (p. 3 of 3)

Annotated script file for Vowel Transcription Exercise 1 ('c:\alvin2\phonetics\vowels\vowelsymbols1.luax'). Comments are shown in italics.

```
    The function below displays the listener's results on the screen. It will run when the task finishes (see 'exp.afterDone').
Also, because the View Results button is enabled when the task begins (see 'exp.addStandardButtons'), results will be displayed
whenever the View Results button is clicked. Note that Courier or some other fixed-width font is needed for the columns to be
aligned in the confusion matrices.

function exp.viewResults()
  local vowels = {"iy", "ih", "ei", "eh", "ae", "ah", "aw", "oa", "oo", "uw", "uh", "er"}
  exp.viewConfusionMatrix{categories=vowels, stimulusColumn=2, responseColumn=5,
    filenameColumn=4, font="Courier", w=80, h=88}
end

The function below will run when the task finishes. The single function that is called displays the listener's results.

function exp.afterDone()
  exp.viewResults()
end

The function below returns 'true' if the response was correct; otherwise, it returns 'false.'

function exp.checkResponse()

  if(exp.response == exp.stimulus[1]) then
    return true
  else
    return false
  end

end
```

Appendix C

Methods for specifying phonetic symbols in the layout() function in Alvin2 and Alvin3 '.luax' script files.

In Alvin2 (Windows only), phonetic symbols must be specified using hexadecimal codes. The hexadecimal values can be found using the *Character Map* utility: (a) in *Character Map*, choose *SILDoulosIPA* (or whatever font is being used), (b) click on the appropriate symbol, and (c) look for the numeric code in the lower left. Note that the font (*SILDoulosIPA*, etc.) needs to be installed in order to use this method to find the numeric codes. The *SILDoulosIPA* and *Phonetics* fonts can be downloaded from http://homepages.wmich.edu/~hillenbr/204.html.

In Alvin3, a far simpler method can be used to input phonetic symbols and any other special characters. The symbols can simply be entered directly into the '.luax' experiment file using any editor that supports Unicode. We use *Text Wrangler* on the Mac and the *Scintilla* editor under Windows. Both are widely available for free download. With the *Scintilla* editor, the encoding (under the *File* menu) needs to be set to "UTF-8" ("UTF-8 Cookie" on at least one version; some trial-and-error testing may be needed to find the right setting). With *Text Wrangler*, the encoding should be set by default to Unicode (UTF-8) when saving the file.

Under Windows, there are a variety of methods for entering Unicode characters. One method is to use an editor such as MS Word that supports Unicode and then copy and paste the desired characters into the '.luax' file. Another approach is to use a utility for Windows called *TypeIt* (http://ipa.typeit.org/), which provides a very simple method for entering phonetic symbols directly from a standard keyboard. The software is available for a modest fee and provides the simplest solution that we have found for entering phonetic symbols in a wide variety of documents.

On OSX, a utility called *Character Viewer* is available, but it may need to be enabled. Under OS X 10.9, go to System Preferences > Keyboard, and click on "Show Keyboard & Character Viewers in menu bar." Character Viewer will appear on a menu in the menu bar in the upper right. After opening it, select "Customize List. . ." from the action button menu, and check the "Phonetic Alphabet" category. These instructions work for OS X 10.9; the method may be different for other versions, so it may be necessary to consult the appropriate Apple Knowledge Base Article for other OS X versions.

Note that, with this method, no specialized phonetic font is needed because the phonetic symbols are included in many widely used fonts that do not require separate installation (e.g., *Arial, Times New Roman, and Calibri*). The line below shows how the phonetic symbols can be displayed for the vowel transcription exercises:

```
button.new{name="iy",    x=5,    label="i",    label2="heed",    value="iy"}
button.new{name="ih",    x=20,   label="ɪ",    label2="hid",     value="ih"}
button.new{name="ei",    x=35,   label="e",    label2="hayed",   value="ei"}
button.new{name="eh",    x=50,   label="ɛ",    label2="head",    value="eh"}
button.new{name="ae",    x=65,   label="æ",    label2="had",     value="ae"}
button.new{name="ah",    x=80,   label="ɑ",    label2="hod",     value="ah"}
```

Appendix D

Alvin results files.

This table shows where Alvin2 results files are stored for each exercise, assuming that the program is installed to the default location ('c:\alvin2'). For example, the first entry is 'c:\alvin2\phonetics\vowels\XXXvsym1.cm'; the 'XXX' refers to the subject identifier that was entered when the exercise was run (e.g., 'jmhvsym1.cm'). These files can be opened and printed using MS Word or Wordpad. Enter the full file name (e.g., 'c:\alvin2\phonetics\vowels\jmhvsym1.cm') at the bottom of the *File Open* dialog box where it says 'File name.' (Note that the place, manner, and voicing exercise produces separate files for each of the three features.) If the columns of the confusion matrices do not line up, do a 'select all' and change the font to Courier. For very wide tables, like the consonant exercises, it may be necessary to adjust the font size and possibly switch to landscape mode before printing the file. Under Alvin3, the default location for output files is the *Documents* directory. For example, under Windows, the output file for the first vowel exercise would be stored as 'c:\Users\(*YourUserName*)\Documents\alvin\vowels\XXXvsym1.cm.' Under Mac OS X, the file would be stored as '/Users/(*YourUserName*)/Documents/alvin/XXXvsym1.cm.'

Vowel transcription training exercise	c:\alvin2\phonetics\vowels\XXXvsym1.cm
Vowel transcription test with feedback	c:\alvin2\phonetics\vowels\XXXvsym2.cm
Word transcription 1 (vowels)	c:\alvin2\phonetics\wordtranscription\XXXwords1.res
Script-a versus open-o (ah-aw) pairs 1 (/hVd/ syllables)	c:\alvin2\phonetics\vowels\XXXahawpairshvd.res
Script-a versus open-o (ah-aw) pairs 2 (words)	c:\alvin2\phonetics\vowels\XXXahawpairs.res
Script-a versus open-o (ah-aw) training	c:\alvin2\phonetics\vowels\XXXahaw1.cm
Script-a versus open-o (ah-aw) test	c:\alvin2\phonetics\vowels\XXXahaw2.cm
Consonant transcription training exercise	c:\alvin2\phonetics\consonants\XXXcsym1.cm
Consonant transcription test with feedback	c:\alvin2\phonetics\consonants\XXXcsym2.cm
Theta versus bar-d training	c:\alvin2\phonetics\consonants\XXXth1.cm
Theta versus bar-d test	c:\alvin2\phonetics\consonants\XXXth2.cm
Diphthong transcription training exercise	c:\alvin2\phonetics\diphthongs\XXXdiph1.cm
Diphthong transcription test with feedback	c:\alvin2\phonetics\diphthongs\XXXdiph2.cm
Word transcription 2 (diphthongs, miscellaneous)	c:\alvin2\phonetics\wordtranscription\words2.res
R diphthong transcription training exercise	c:\alvin2\phonetics\diphthongs\XXXrdiph1.cm
R diphthong transcription test with feedback	c:\alvin2\phonetics\diphthongs\XXXrdiph2.cm
Place, manner, and voicing exercise	c:\alvin2\phonetics\consonants\XXXpmv.cmp (.cmp=place) c:\alvin2\phonetics\consonants\pmv.cmm (.cmm=manner) c:\alvin2\phonetics\consonants\pmp.cmv (.cmv=voicing)
Reverse phonetic transcription 1	c:\alvin2\phonetics\reverse\XXXrev1.res
Reverse phonetic transcription 2	c:\alvin2\phonetics\reverse\XXXrev2.res
Reverse phonetic transcription 3	c:\alvin2\phonetics\reverse\XXXrev3.res
Word transcription: schwa	c:\alvin2\phonetics\schwa\XXXschw1.res
Word transcription 3 (general practice)	c:\alvin2\phonetics\wordtranscription\XXXwords3.res
Word transcription 4 (general practice	c:\alvin2\phonetics\wordtranscription\XXXwords4.res
Word transcription 5 (general practice)	c:\alvin2\phonetics\wordtranscription\XXXwords5.res
Word transcription 6 (general practice)	c:\alvin2\phonetics\wordtranscription\XXXwords6.res