The use of Praat in corpus research

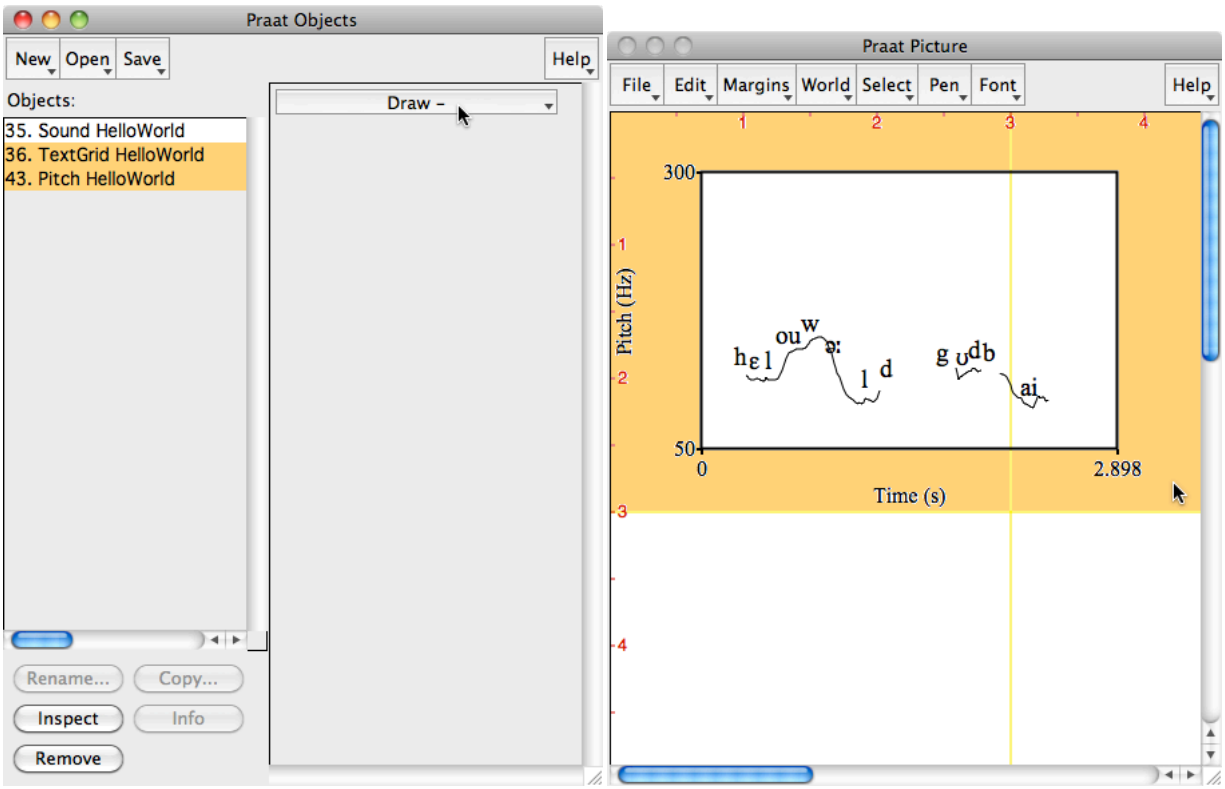
Paul Boersma

Praat is a computer program for analysing, synthesizing and manipulating speech and other sounds, and for creating publication-quality graphics. It is open source, and available free of charge for all major computer platforms (MacOS, Windows, Linux), on both 32-bit and 64-bit operating systems. It can be downloaded from praat.org.

A speech corpus typically consists of a set of sound files, each of which is paired with an annotation file, and metadata information. Praat's strengths are in the acoustic analysis of the individual sounds, in the annotation of these sounds, and in browsing multiple sound and annotation files across the corpus. Corpuswide acoustic analyses, leading to tables ready for statistical analysis, can be performed by scripting, as described by Caren Brinckmann elsewhere in this book.

1. The windows of Praat

When you start up the Praat program, two windows pop up: the Object window at the left, and the Picture window at the right. This figure shows both windows after the user has worked for some time with Praat:



At the left side of the figure we see the Object window, which here contains three objects: a Sound (waveform) object, a TextGrid (annotation) object, and a Pitch (fundamental frequency) object. The TextGrid and Pitch objects are selected. To the right of the object list we see the commands available for the current selection; apparently, when you select a TextGrid and a Pitch together, you can only choose from a menu with drawing commands. At the right side of the figure we see the Picture window, which here contains a drawing of a TextGrid (a phonetic annotation of the utterance *Hello world. Goodbye.*) superposed on the pitch contour of this same utterance. Apparently, the user has just executed one of the **Draw** commands from the menu in the Objects window, and the result is this drawing in the Picture window. The orange rectangle around the drawing demarcates the selected part of the Picture window. If the user now does **Copy** in the Picture window and then **Paste** in her journal article that she opened in her favourite word processor program, the annotated pitch curve will show up in her article.

1.1 The Object window

A question that is often asked about Praat is why the Object window is necessary: couldn't Praat, when the user opens a sound file, just show that file in its own sound window, and nothing else, as happens in some other programs? My answer is that the world of speech analysis is too large for that.

The world of speech analysis is a world consisting of sounds, spectra, pitches, annotation, speakers, lips, muscles, processing models, grammars, data tables, and much more. First, this means that a computer program that describes this world cannot get by by showing only a sound in a sound window: there also have to be spectrum windows, pitch windows, annotation windows, muscle windows, grammar windows, table windows, and more. Praat has all of these windows, but even they do not suffice, because several actions that phoneticians like to take require *two* or even *three* objects. In the figure above, for instance, the user drew a combination of an annotation and a pitch curve into the Picture window, and to achieve this she first had to select a TextGrid object and a Pitch object together. While it would be possible to draw a TextGrid from a TextGrid window, or a Pitch from a Pitch window (both possibilities exist in Praat), it is not possible according to any established graphical user interfaces to draw from two windows at a time. That is, to draw an annotation and a pitch curve together, the user would have to bring both the TextGrid window and the Pitch window to the top, and then say **Draw**. Since there are no intuitive ways of handling such situations in a one-object-per-window user interface, Praat uses the Object window for listing its objects, so that the user can select multiple objects at a time. Such multiple selection is natural not only in the above TextGrid-and-Pitch case, but also for instance when you want to concatenate 10 sounds: you just select the 10 Sound objects in the list and choose **Concatenate**. I would know of no intuitive way to accomplish this if every sound were only visible in its own window.

The Object window exists, then, because in the large world of speech analysis you will often want to work with *combinations of objects* (e.g. a TextGrid and a Pitch), with *multiple objects* of the same type (e.g. the 10 Sounds to be concatenated), and generally with a *large*

number of objects: for instance, extracting the 64 channels from an EEG object gives you 64 Sound objects in the list, and then choosing **To Spectrum...** gives you the corresponding 64 Spectrum objects. Or, dropping all 40 sound files and all 40 annotation files in a certain directory on the Praat icon makes Praat loading all of them into the object list for fast inspection. Summarizing, having the Object window gives you the benefits of flexibility and scale.

Given the existence of an Object window, it is the natural place to handle file input and output. The objects in the list are just data structures in memory, like e.g. the documents in a word processor, and they are not necessarily connected to a file on disk. Thus, the Object contains the New menu, with which you can create new objects from scratch, the Open menu, with which you can read sound files, pitch files, annotation files, grammar files, and so on, from disk, and the Save menu, with which you can write any object (or collection of objects, if you select more than one of them) to disk in various formats.

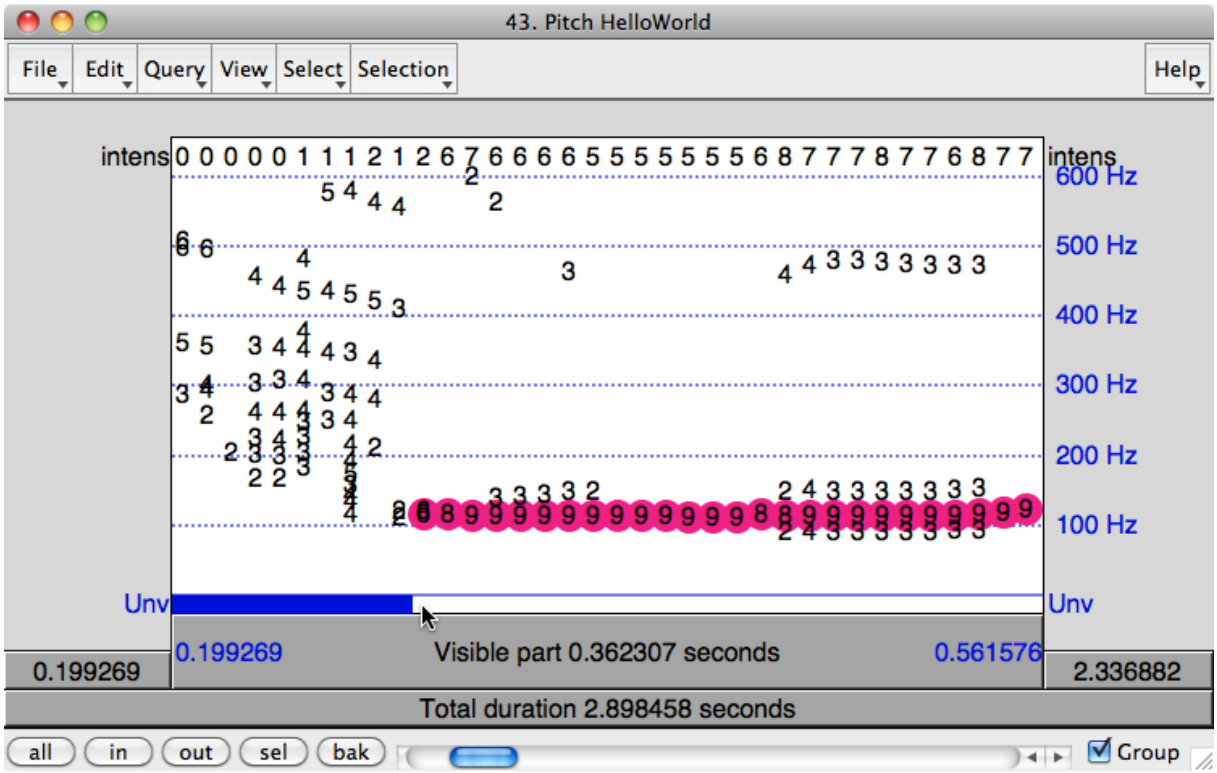
1.2 The Picture window

As with the Object window, a question that is often asked is why the Picture window is necessary: couldn't Praat just support copy-paste from the Sound window and the other object-specific windows? My answer is that that would not lead to publication-quality drawings. The Sound window is optimized for *viewing* (inspecting) and *editing* (modifying) waveforms, and the Pitch window does the same for pitch curves. Viewing and editing require seeing interpolated sample values, having a cursor and selection markers, having thick red pitch dots that you can click on and grab, and so on. Publishing requires a representation of waveforms and pitch curves as thin, typically black-on-white lines, with times and frequency markers at regular distances along the edges, and texts like "Time (s)" and "Pitch (Hz)" along the horizontal and vertical axes, as in the figure above. According to the What-You-See-Is-What-You-Get (WYSIWYG) principle known from the human-computer interface literature, this publication-level representation should be visible to the user at the moment she chooses **Copy** or **Print**. If this representation cannot be in the Sound or Pitch window, it has to be in a different window, and that is what the Picture window is for.

Given the existence of a Picture window, it is the natural place to receive all drawings that move outside Praat. There are **Draw** commands in the Objects window if you select a Sound, Pitch, TextGrid, or some other object, and there are **Draw** commands in the Sound window, the Pitch window, or the TextGrid window, which allow you to draw the visible parts of the viewed objects into the Picture window. The Picture window then allows you to add markers and legends in the margins of the drawing, and to add text, circles, arrows and the like inside the drawing, at locations specified in seconds and Hz. From the Picture window you can copy, print, or save the drawings. The Picture window supports any colour, font size, line width, and line type, several fonts, and a large number of international and phonetic characters.

1.3 Viewer and editor windows

As said, viewing and editing windows for specific objects are optimized for viewing and editing. Such a window typically comes to the screen when you select an object and click **View & Edit**. For instance, if you select a Pitch object and click **View & Edit**, a Pitch window comes to the screen. When you zoom in a bit, it can look like follows:

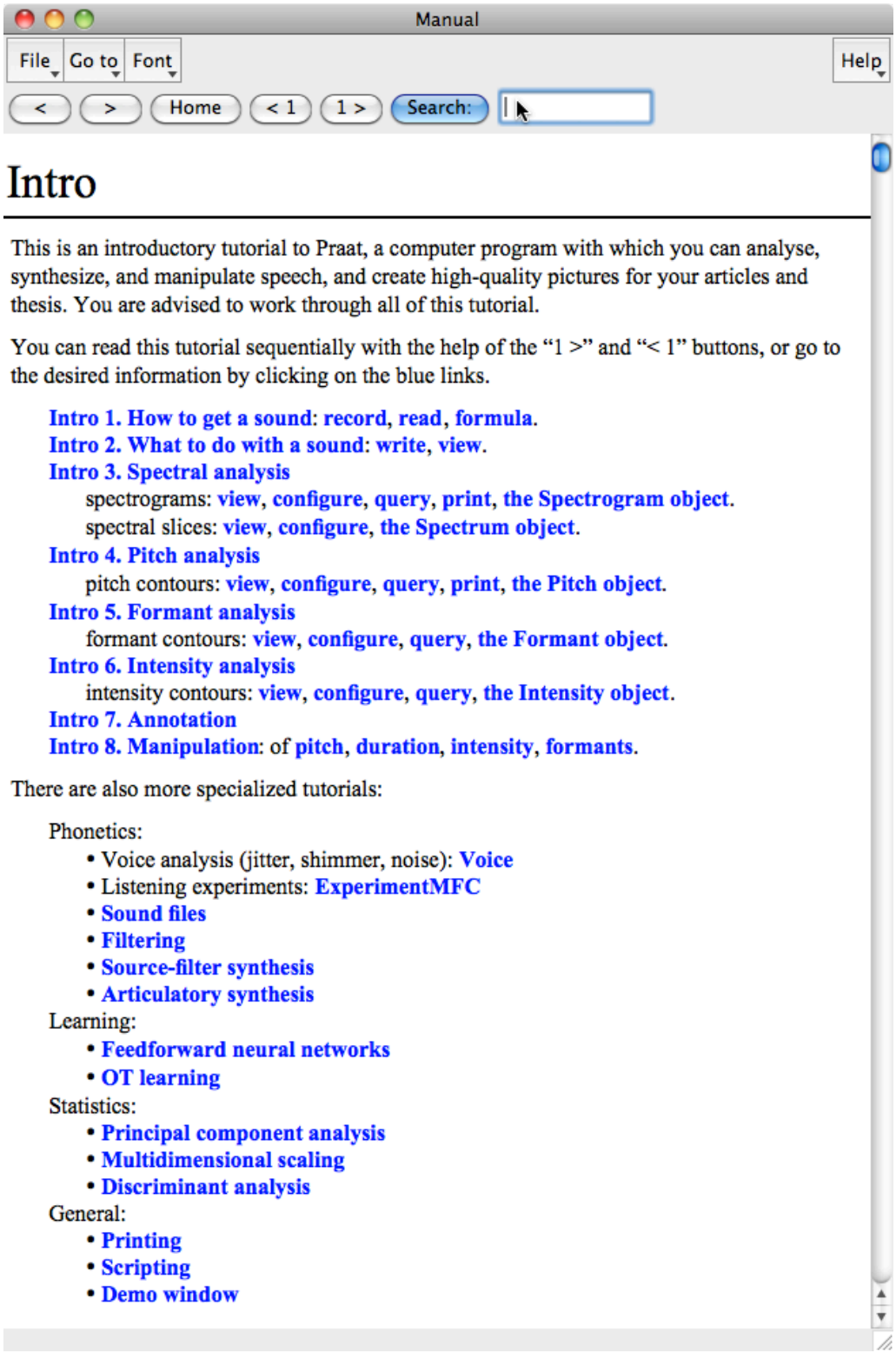


This Pitch window contains numbers between 1 and 9 to depict the quality of separate pitch candidates, red dots to mark the pitch candidates that are in the current optimal path, and a blue “unvoiced” bar at the bottom. You can click on the numbers and on the unvoiced bar to modify the pitch curve. Please note that the representation of the Pitch object in this window is quite different from its representation in the Picture window above.

Many object types in Praat can be viewed and edited in their own windows. Further on in this chapter we see examples of the Sound window, the TextGrid window, and the Corpus window.

1.4 Other windows

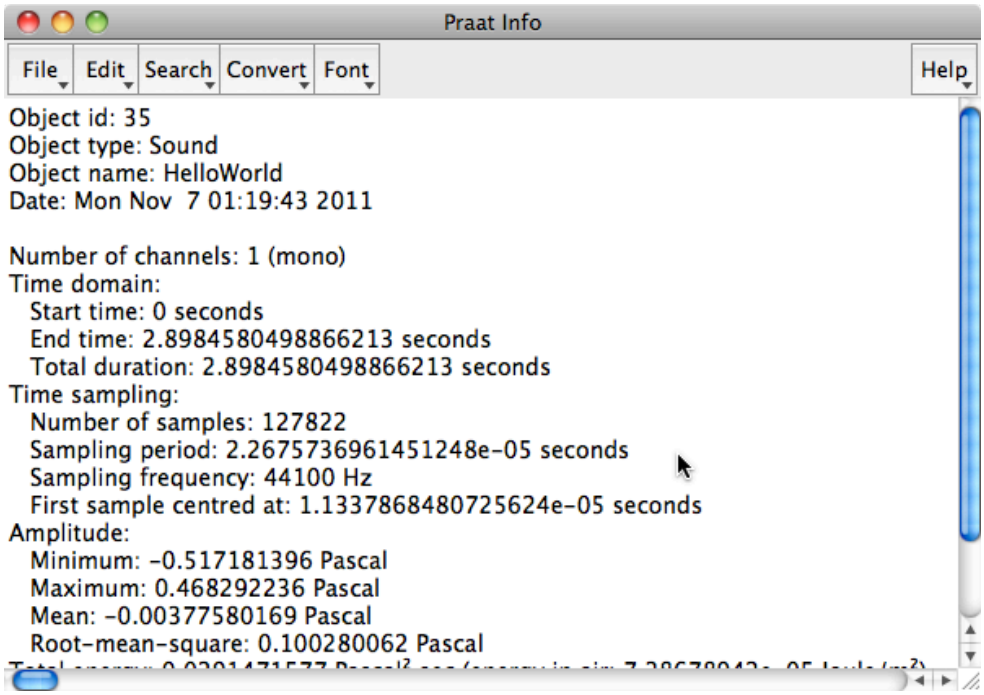
An important window in Praat is the manual window. When you choose **Intro** from a Help menu, the following manual window pops up:



The Intro page is the best location for beginning users of Praat to start to get to know the program. As we see in the figure, the Intro contains information about the basic handling of

sounds, spectra, pitches, annotation and manipulation, as well as links to more specialized tutorials. The blue texts in the Praat manual take you to other pages; to read a tutorial sequentially you can use the "1 >" button. There are currently around 2000 manual pages, whose levels vary between tutorials, reference pages, and technical information. The **Search** field at the top of the manual window gives you a list of pages matching the texts you type there; for instance, type "sound pitch" and click the **Search** button to get to the pages that discuss how a Sound object can be converted to a Pitch object, or to get to the pages that discuss what you can do when you select a Sound object and a Pitch object together.

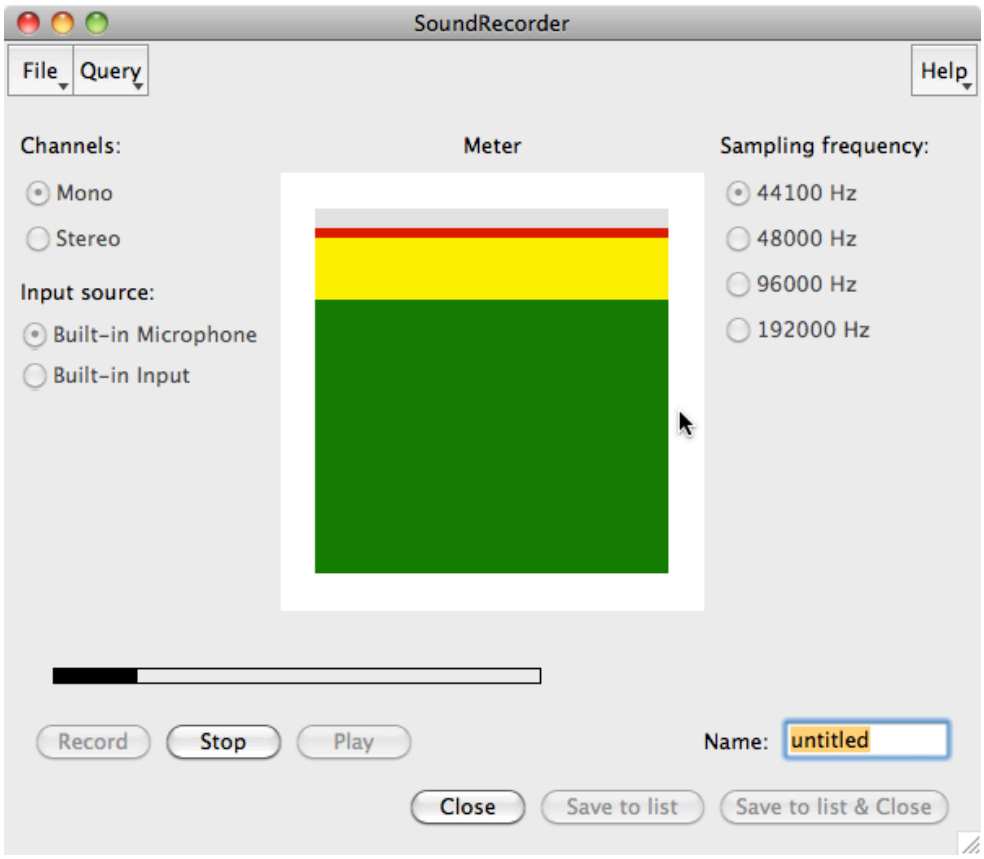
The last window I discuss in this introduction is the Info window. If you select the Sound object in the first figure above and click the **Info** button in the Object window, Praat writes some information about the Sound object to the Info window:



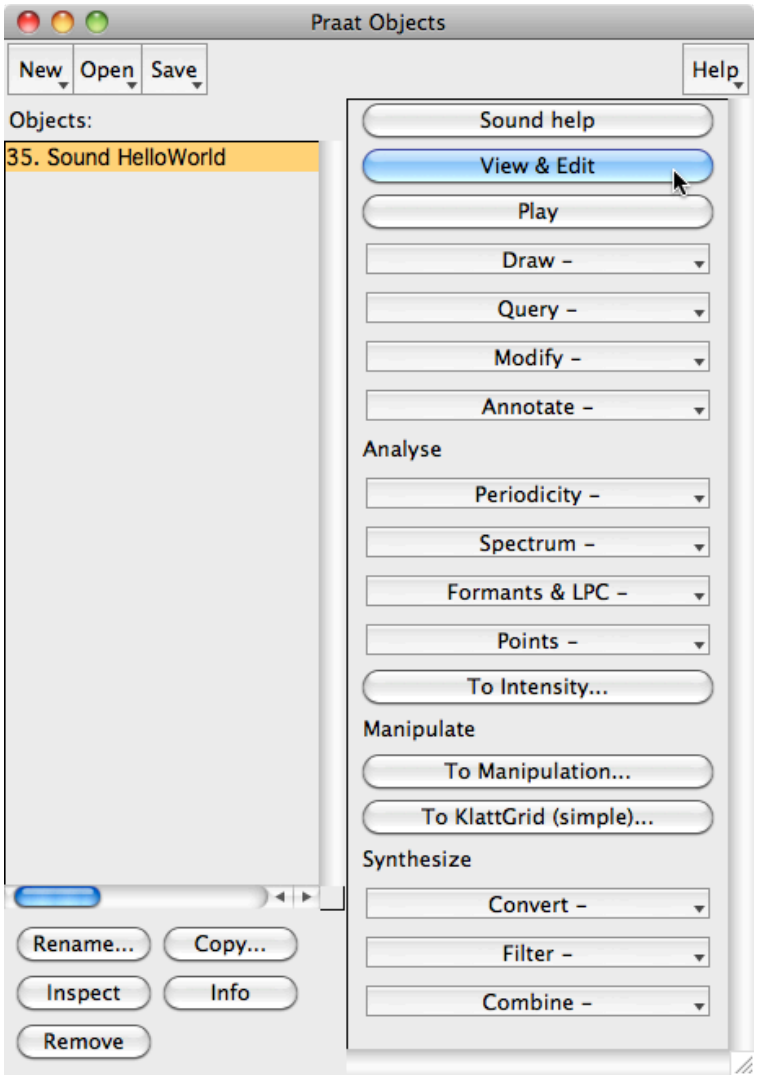
Results from any commands in the Query menus, which answer questions about a selected object, are also written into the Info window. The contents of the Info window can be copy-pasted elsewhere, and saved to disk.

2. The Sound window

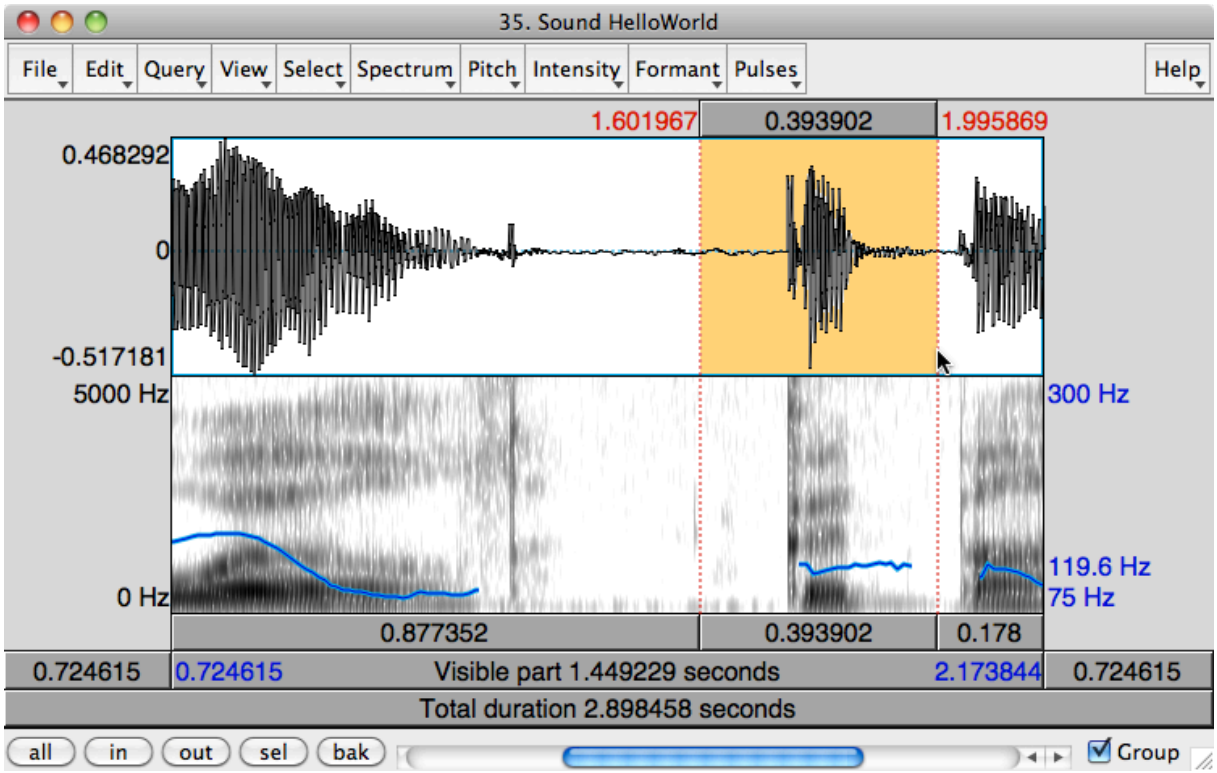
The first thing you want to do when you start up Praat for the first time is to put a Sound object in the list. To put an existing sound file in Praat's object list, you can drop it on the Praat icon, or use **Read from file...** from the Open menu. A possibly more interesting way to get a Sound is to choose **Record Sound...** from the New menu, which pops up Praat's SoundRecorder window:



In this window, you press **Record**, then you speak, then you press **Stop**. You can then listen to your recording with **Play**, and if the result is OK, you get the sound a name, such as "HelloWorld" and click **Save to list & Close**. The recording then shows up as a new Sound object in the list:



In the menus that appear to the right of the object list you see all the things you can do to the Sound: playing, drawing, querying, modifying, annotating, analysing, manipulating, converting, filtering, and combining it with other Sounds (if these are selected as well). The most "attractive" button, however, says **View & Edit**, and if you click it (or just press the Enter key), the selected Sound pops up in a new Sound window:



In this example, we see the waveform of the Sound at the top (with time running from left to right), two analyses (a pitch superposed on a spectrogram) below it, and eight buttonlike rectangles with numbers above the waveform and below the analyses. These rectangles show what part of the Sound is visible, what part is selected, and what the remaining parts are. Since I already zoomed in once (by clicking the **in** button), only half of the total duration is visible, and one quarter of the Sound (0.724615 seconds) lies before (to the left of) the visible part, and one quarter lies after it. Since I selected a stretch of 0.393902 seconds, the visible part divides up into three parts. To play any of these parts, you click on the corresponding rectangle (to play the selected part, you can also just press the Tab key). You can scroll through the sound as you are used to in other computer programs such as word processors and web browsers.

The menus at the top of the window determine which analyses are visible. The grey-scale image below the waveform is the *spectrogram*, a representation of the frequency content of the waveform at each point in time. The viewing range has been set here from 0 to 5000

Hz, which allows you to see the formant movements (e.g. at the beginning the F2 rises from a [w] position to a [ə] position). The present chapter has no room to dwell on the large usefulness of the spectogram for phonetic research; for that, see Ladefoged (2005) and Ladefoged and Maddieson (1996).

The blue curve superimposed on the spectrogram is the *pitch curve*, a representation of the periodicity in the waveform at each point in time. The viewing range has been set here from 75 to 300 Hz, which is a typical range for male speech. At the beginning we can see the falling tone on *world*.

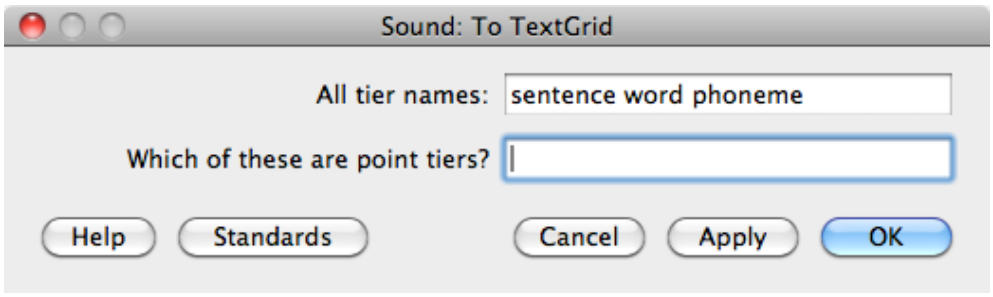
Other analyses that can be visualized in the Sound window are automatically measured formants, an intensity curve, and the glottal pulses. You can extract all these analyses from the Sound window to the Object window as separate Spectrogram, Pitch, Formant, Intensity, or PointProcess objects, if you want to inspect, edit, save, or draw them in more detail than the Sound window can do. Finally, the Pulses menu contains a **Voice report** command, which reports the voicedness, jitter, shimmer, and harmonics-to-noise ratio of the selected part of the sound.

3. Annotation with the TextGrid window

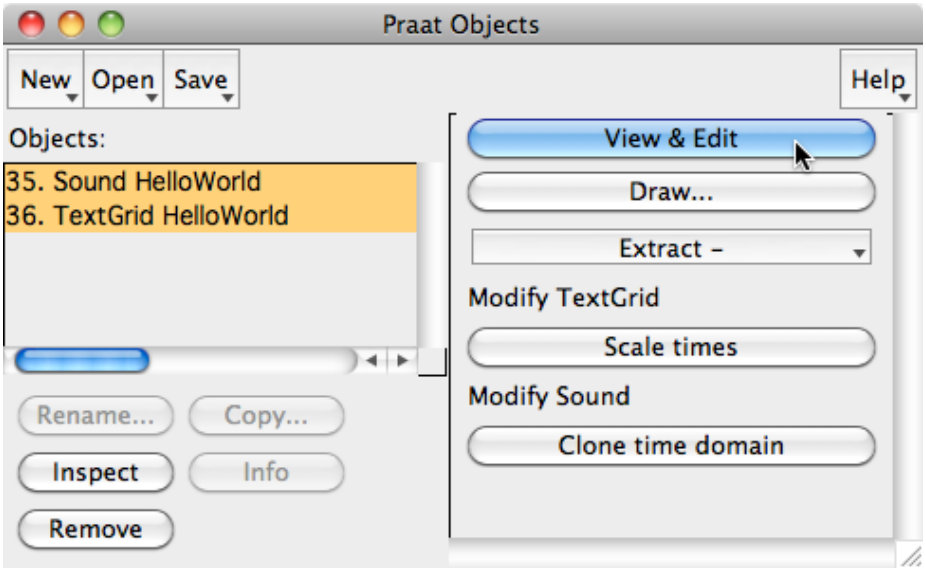
In Praat, you can *annotate* (or *segment*, or *label*) several types of time-based signals. The most often annotated type is the Sound.

The object type that handles annotation is the TextGrid. A TextGrid is a collection of *tiers* (this rhymes with *cheers*, not with *liars*). A tier is an ordered sequence of texts, each of which is connected to a point in time or to a stretch of time, as explained below.

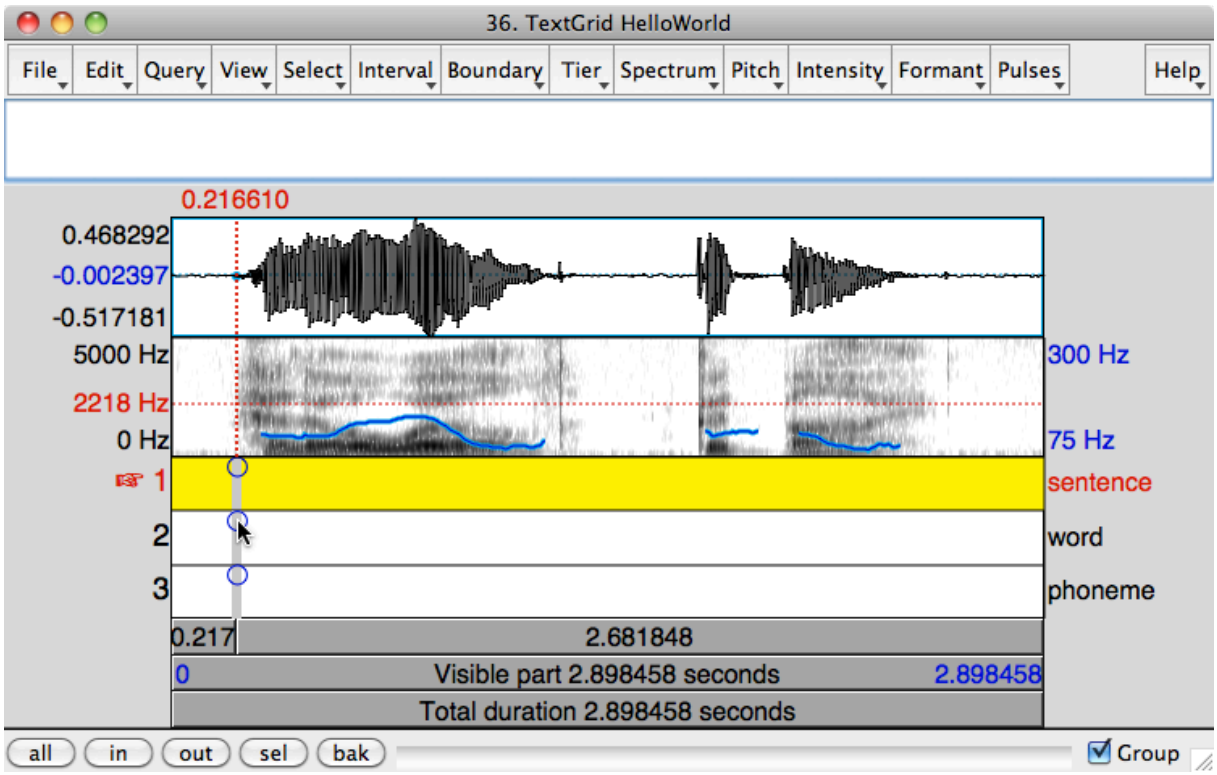
To start annotating a Sound, you select a Sound object and choose **To TextGrid...** from the **Annotate** menu. A TextGrid creation window appears, which allows you to specify the names of the tiers:



Suppose you want to annotate sentences, words, and phonemes. In the TextGrid creation window you may then want to specify three tiers, named "sentence", "word", and "phoneme". You achieve this by typing "sentence word phoneme" into the text field labelled **All tier names**, as in the figure above (if you don't yet know whether you need a certain tier or not, or whether it has the correct name, or whether the order of the tiers is correct, don't worry: you can always add, remove, rename and reshuffle tiers later). You can leave the second text field empty, as in the figure, and click **OK**. A new TextGrid object, with the same name as the original Sound, then appears in the object list. You then select the Sound and TextGrid together and choose **View & Edit**:



A TextGrid window then appears on the screen:

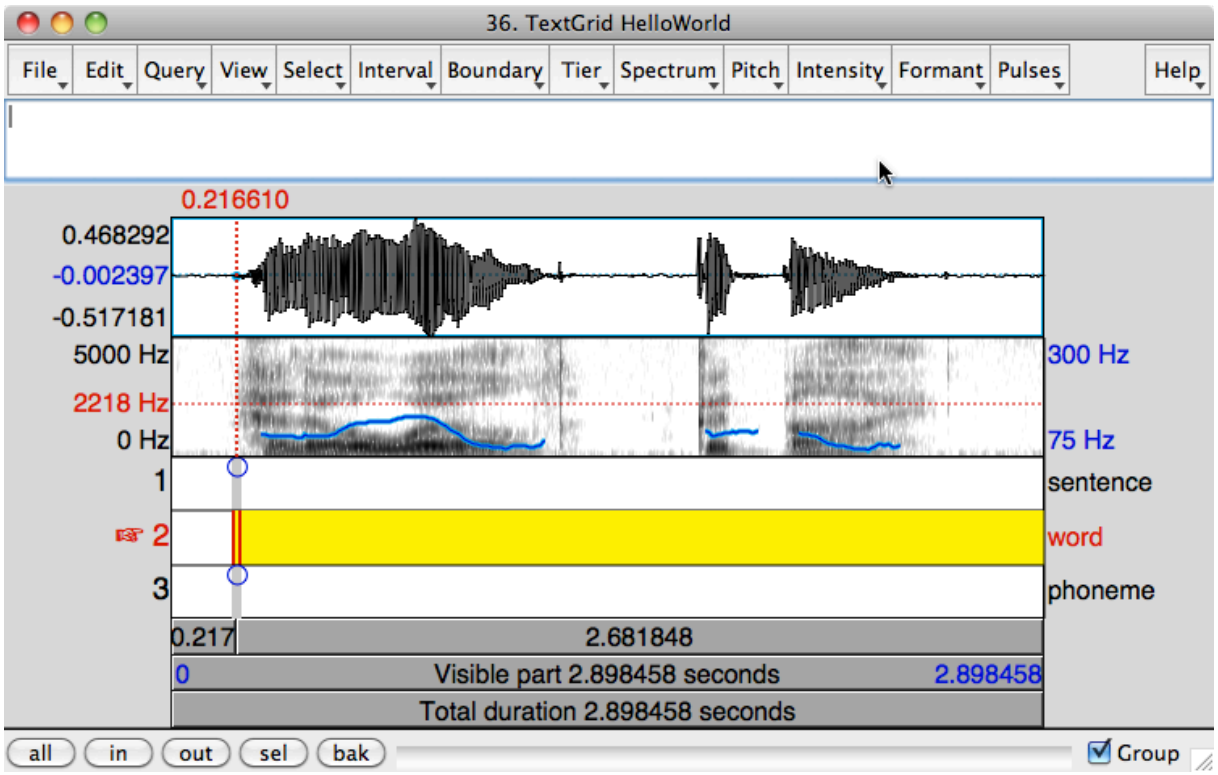


This TextGrid window shows three things: at the bottom, the contents of the TextGrid, which is initially just three empty tiers; at the top, the waveform of a copy of the Sound; in the middle, some analyses (in the figure, a spectrogram and a pitch contour) of this copy of the Sound.

In a TextGrid you can have two kinds of tiers: interval tiers, in which you can annotate stretches of time, and point tiers, in which you can annotate time points.

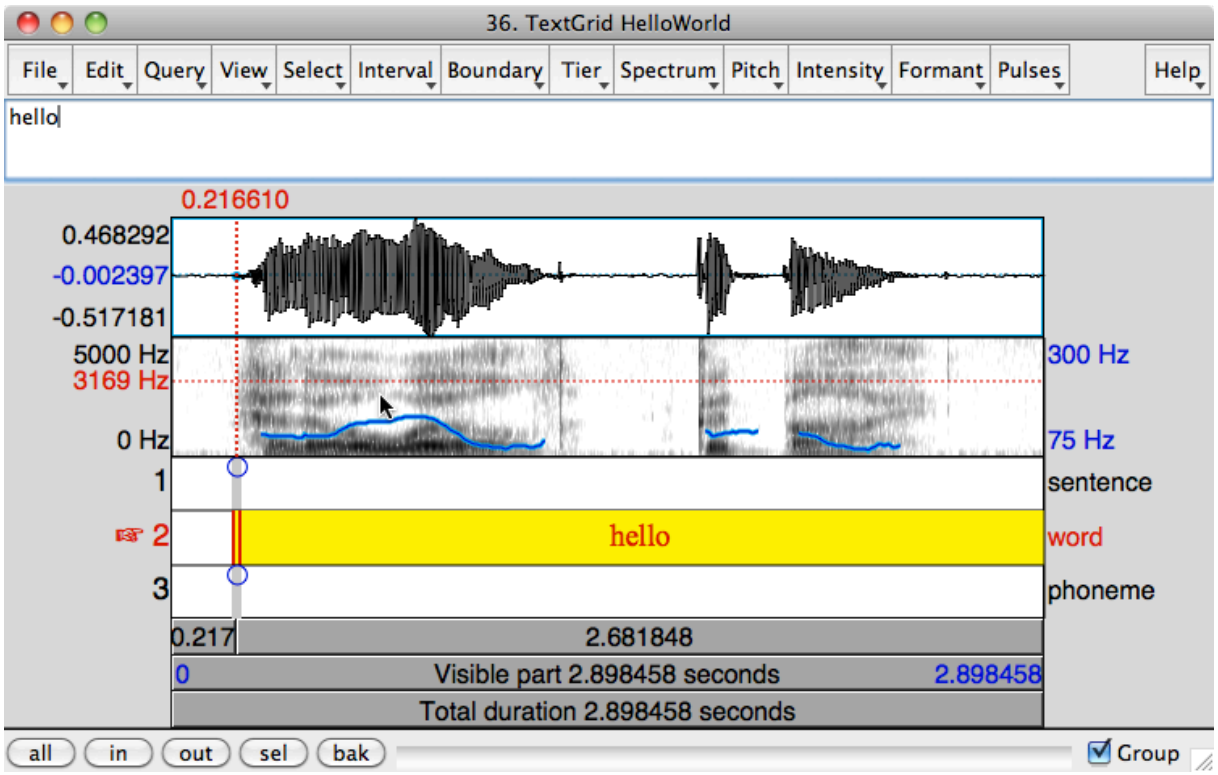
3.1 Interval tiers

All three tiers in the example TextGrid above are *interval tiers*. An interval tier is a sequence of adjacent time stretches that you can annotate. In the figure above, for instance, each tier consists of only one *interval*, which runs from 0 to 2.898458 seconds, and this interval is *empty*, i.e. it contains the text "". In the figure, the user has already clicked in the spectrogram, so that there is a cursor at 0.216610 seconds. This cursor extends through the three tiers as a thick grey line, which comes with a blue circle on each tier. In the figure, the user has moved the mouse pointer onto the blue circle on the second tier. If the user now clicks in this circle, a new *boundary* will appear at the cursor position on the second tier:



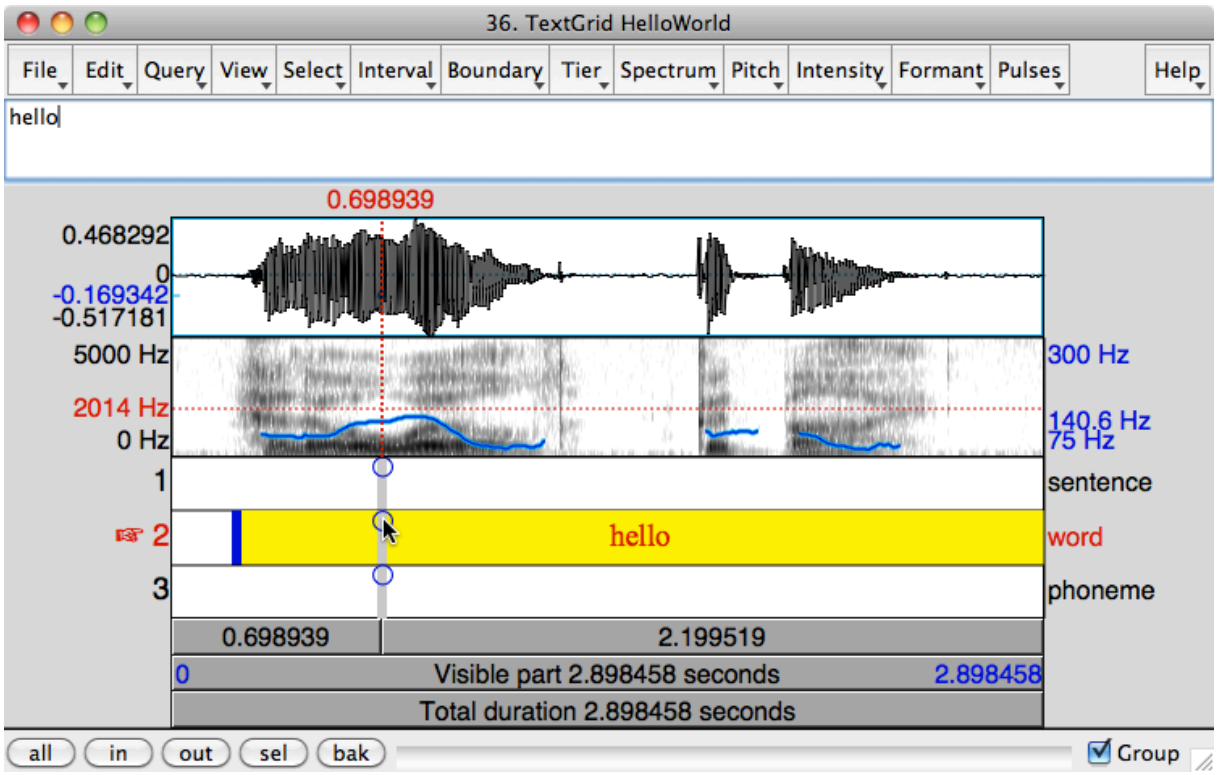
In this figure we see that now the second tier is currently selected (as seen by the red pointing finger, and the red colour of the tier number and tier name), and that inside this tier the new boundary is selected (as seen by the red vertical line with the yellow vertical line through its middle) and the second interval, which contains the cursor time (at its left edge) is also selected (as seen by the yellow rectangle).

You can directly type text into the selected interval. For instance, if you type “hello”, the second interval will come to contain the text “hello”:

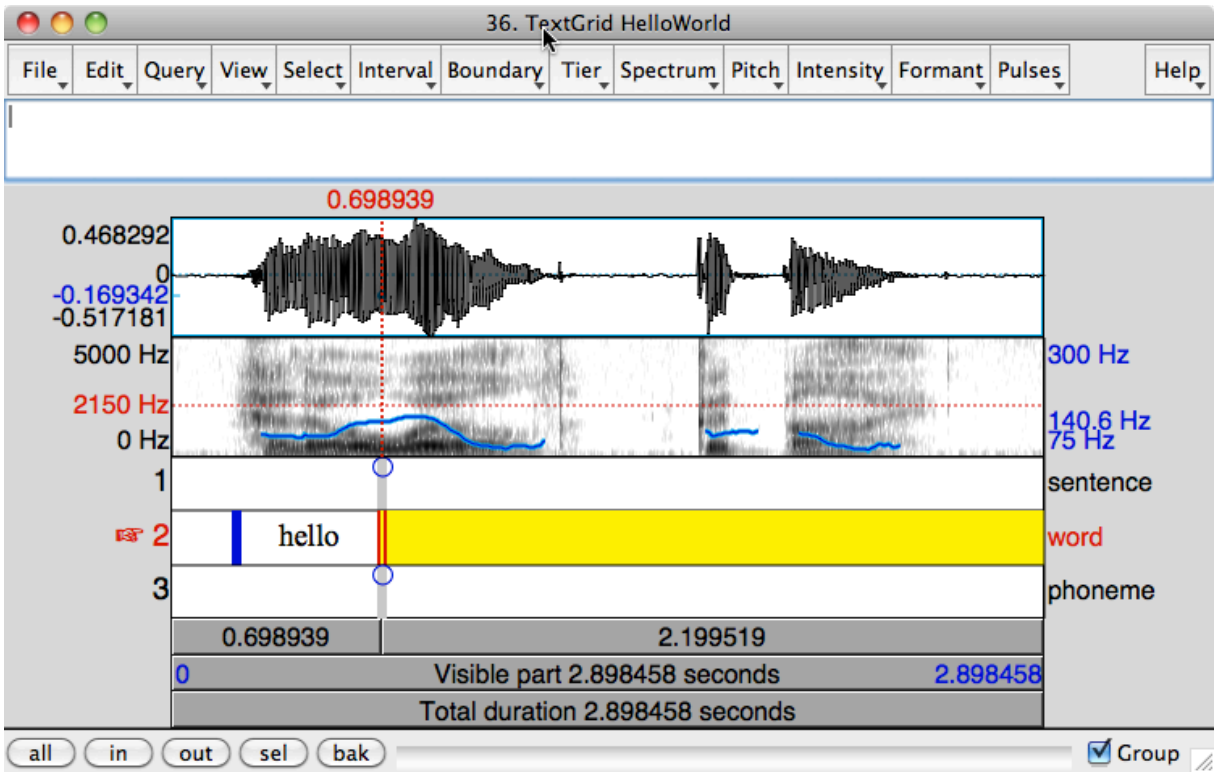


This figure shows that the text that you type will show up in the text field above the waveform, as well as in the selected interval (the reason for this is explained below).

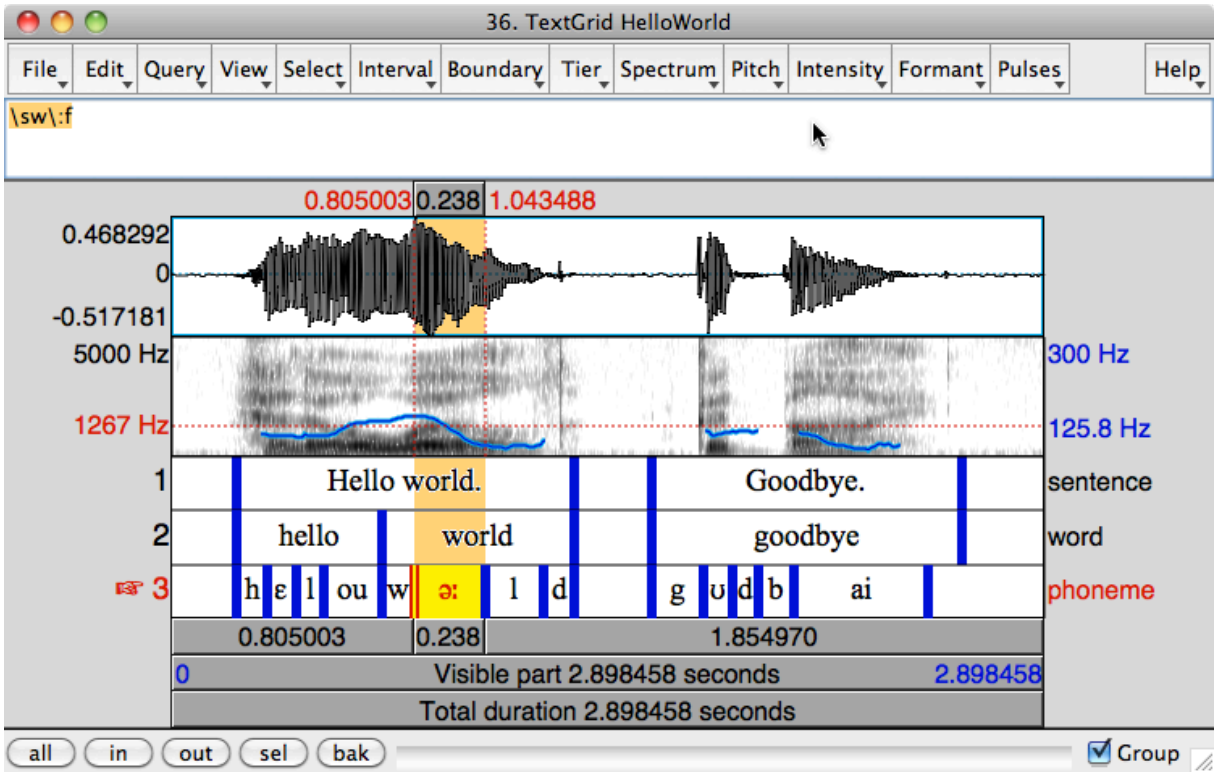
Next, you want to add a boundary at the end of the word “hello”, and in the figure above the mouse pointer is already at the right location in the spectrogram, at 0.698939 seconds. When you click there, the result is:



Since the cursor time is still in the second interval (of the second tier), this interval is still selected. When you now click the blue circle (the mouse pointer in the figure already points at it), a second boundary appears, and the text “hello” moves to the left of that boundary, because the text cursor in the text field is after the last letter of “hello” (in the figure above). Instead of clicking into the blue circle, you can also just press the Enter key, which creates a new boundary at the cursor time in the selected tier. Either way, the result is that the new boundary is selected and you are ready to type text into the third interval:



You can go on like this, and after editing all three tiers, perhaps with the help of some zooming and scrolling, the result may look like follows:



Every tier has now been filled with data. There are now nine empty intervals, and all other intervals contain one or more characters. The selected interval contains the text "əː", which could have been typed directly as "əː" with a Unicode input method but has here been typed quickly by an ASCII keyboard as a sequence of two backslash trigraphs, namely "\sw\:f" (= "schwa + phonetic colon"), which is seen in the text field and which is how the data will be saved to disk once you decide to save the TextGrid (there is a button for converting all backslash trigraphs in a TextGrid to Unicode if you want the TextGrid file to be readable outside Praat).

Direct manipulation. Some types of annotation are very fast: if you annotate one tier from left to right, then it is typically a sequence of click (in the spectrogram, to set the cursor time), type Enter (to add a boundary in the tier), type the label, and then again click-Enter-type and so on. You don't need to double-click to open a data entry window, and you don't need to press command-enter to "save" the text into the object: the changes will occur at the moment you type. In the literature on human-computer interaction, this is called *direct manipulation*, a term that stresses two ideas at the same time, namely that you change things by grabbing them or typing, and that these changes are effective without confirmation.

The concept of direct manipulation can be observed by the strong connection of the TextGrid window to the TextGrid object in the list: if you type a character into a tier in the TextGrid window, the TextGrid object in the list immediately changes. You can see this if you open a TextGrid in a second window (i.e. choose **View & Edit** a second time): changes you make in one window are immediately reflected in the other.

Deliberately, the directness of the user interface applies to objects only, not to files. That is, a TextGrid is changed easily by dragging boundaries, clicking in circles, and typing text, but its state on disk is not changed until you explicitly perform one of the **Save** commands to create or overwrite a TextGrid file on disk. This combination of direct manipulation of objects and explicit saving to disk allows you to edit the data fast, and to experiment with the data easily (helped by the **Undo** command), without fear of overwriting a precious piece of saved data inadvertently. In these respects, Praat works quite differently

from several other speech annotation programs, and instead is more similar to most word processors and drawing programs.

Selecting a boundary. You can select a boundary by clicking in its vicinity. That is, if you click at the time of 1.065 seconds in the third tier of the figure above, the cursor will move to the exact end time of the interval labelled "əː", i.e. to 1.043488 seconds (in reality, the precision is even greater than a microsecond). The precision of this "snapping" is perfect, e.g. much smaller than a pixel.

Changing the position of a boundary. To change the position of a boundary, just grab it and drag it about. You can drag the boundary to any other time position as long as you don't try to move it across its left or right neighbour. To move a boundary to the cursor time, just drag it to the vicinity of the cursor line and it will snap exactly to the cursor time.

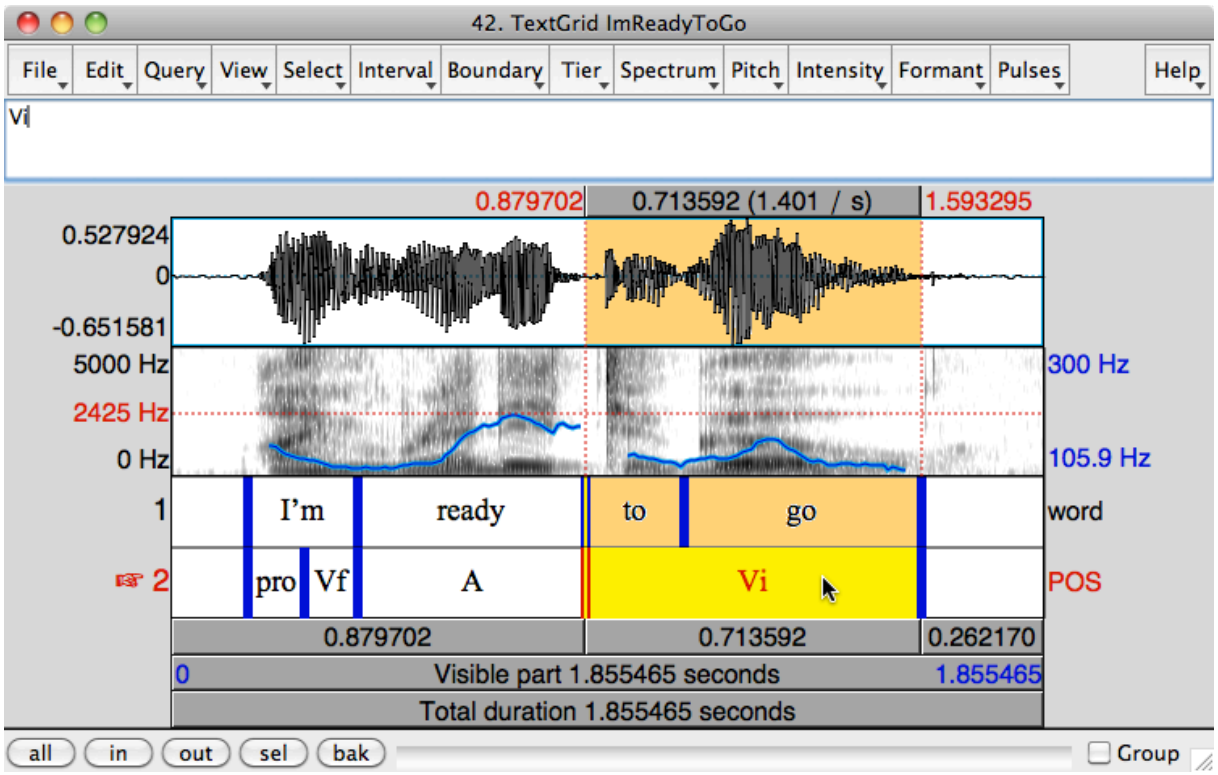
Perfectly adjacent intervals. In the figure, the interval labelled "hello" has to be perfectly adjacent to the interval labelled "world". Since the tier is entirely divided up in intervals, with no empty spaces in between (an empty interval is still an interval), this adjacency is fully automatic: each boundary creation creates two perfectly adjacent intervals, i.e. there are never gaps between intervals.

Adding two boundaries at a time. In the figure you could have created the interval "Hello world." by selecting this whole sentence in the spectrogram and then pressing the Enter key. This adds two boundaries: one at the start of the selection and one at the end of the selection.

Alignment across tiers. There are several methods for making two boundaries on different tiers perfectly aligned. In the figure above, the left boundary of "Hello world." could have been set by clicking on the left boundary of "hello", then clicking in the circle that appears in the first tier. Or, if you want the right boundary of "ai" to be at the exact same time location as the right boundary of "goodbye", you can drag the right boundary of "ai" from the third tier to

the vicinity of the right boundary of "goodbye" *on the second tier*. The boundary will then snap to the exact same time location as the end of "goodbye". This perfect alignment across tiers can be seen in the TextGrid window: for instance, if you click on the left boundary of "goodbye" in the figure, it will be selected and the yellow vertical line will also run through the left boundary of "Goodbye." on the first tier and the left boundary of "g" on the third tier. Finally, if you drag a boundary to move it and at the same time hold the Shift key pressed, any boundaries perfectly aligned with this boundary on other tiers will move with it, thus preserving the existing alignment (there is a preference setting for exchanging the dragging and Shift-dragging behaviours).

Tier hierarchies. Several other annotation programs allow tiers to be hierarchically organized. The most common case is hierarchical subdivision, such as words that consist of syllables, syllables that consist of phonological segments, and so on. The most advertised example is that of the part of speech (POS), which allegedly has a one-to-one relationship with the word. In my opinion, the importance of such hierarchies is overrated. In the languages of the world, syllables often cross word boundaries (as in French *petit ami*, which is syllabified as .pə.ti.ta.mi.), segments can cross syllable boundaries (as in Italian .grap.pa., with p.p being a single segment), and even the relationship between word and POS is not clearly one-to-one. Consider, for example, the following TextGrid:



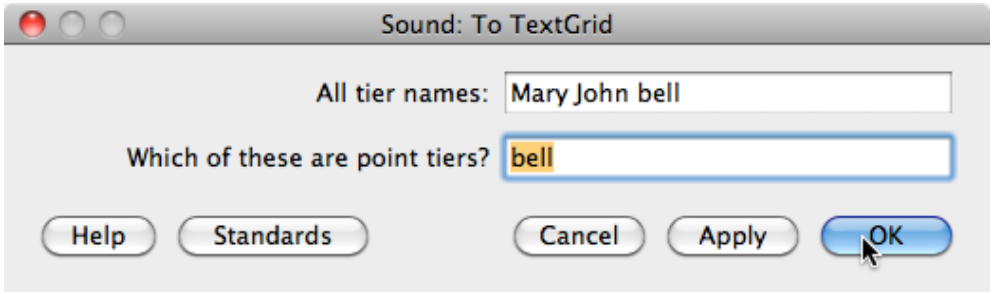
Here, the definition of a word is the easiest one: anything separated in the orthography by spaces or punctuation marks. The word "I'm", then, corresponds to two parts of speech, namely a pronoun and an inflected (finite) verb, and, conversely, the infinitive verb corresponds to the two words "to" and "go". Of course, you can adapt the definition of what a word is to your syntactic theory, or adapt the theory to your definition of a word, but that would either force you to reject your easy, and perhaps automatic, orthographic definition of the word or to reject your well-argued-for theory of what a syntactic word is. Praat, then, chooses not to restrict the relationships between boundaries on different tiers. To help you maintain a near-hierarchical relationship between tiers, Praat provides the easy cross-tier alignment methods described in the previous paragraph, and also has commands for searching for boundaries that are on tier X (e.g. the syllable tier) but not on tier Y (e.g. the segment tier), thus quickly finding locations that violate an approximate hierarchy (e.g. all the geminates).

I have to admit here that Praat's TextGrid model, with its free cross-tier alignment, still has the restriction that intervals cannot be discontinuous. Thus, in a split infinitive like *to boldly go* you cannot label *to* and *go* as the same Vi, and *boldly* separately as an ADV. For such cases, which are much more common in many languages other than English, the

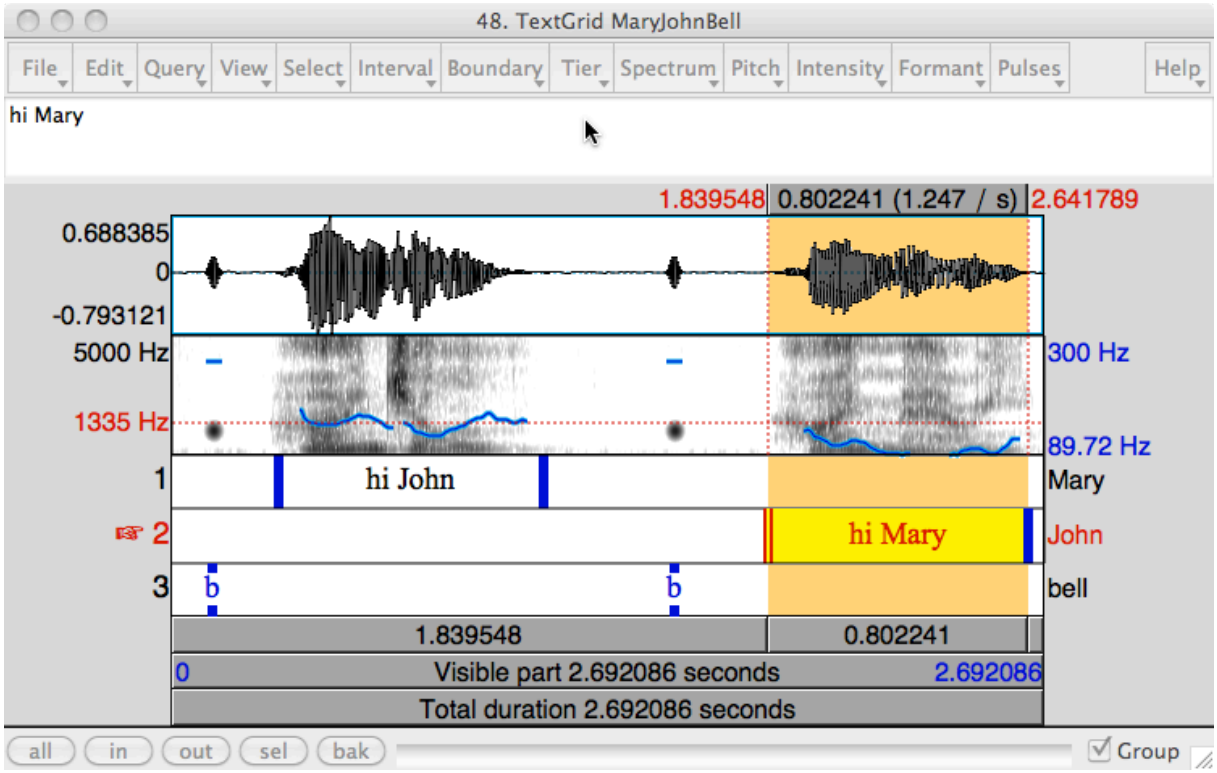
TextGrid model fails, as do its hierarchical counterparts in other programs. The "annotation graph" model (Bird and Liberman, 2001) can represent such situations, because it allows overlapping intervals on its "tiers", but its more restrictive implementation in EXMARaLDA (Kipp, this book) cannot; the annotation graph model allows even more flexibility by providing a "class" property for each label, by which coreference between intervals on the same or different tiers can be represented; none of the annotation programs described in this book seem to have built-in support for this capability.

3.2 Point tiers

In the examples above, the TextGrid annotates interval, i.e. stretches of time. You can also have tiers in which *points* in time are annotated. To have a TextGrid for two speakers, Mary and John, who have to speak after waiting for a bell to chime, you can select a Sound, choose **To TextGrid...**, and specify the three tiers in this way:



In the first text field, you supply the three tier names, together with their order ("Mary John bell"); in the second text field, you tell Praat which of these three tiers has to be a point tier ("bell"). When you then click **OK**, select the new TextGrid together with the original Sound, and choose **View & Edit**, a TextGrid window pops up with two interval tiers and one point tier:



In the figure the three tiers have already been edited. On the third tier, two points have been inserted, and each of these was labelled “b”. These points can be handled in much the same way as the boundaries of the intervals can be handled, except that you can drag points beyond their neighbours.

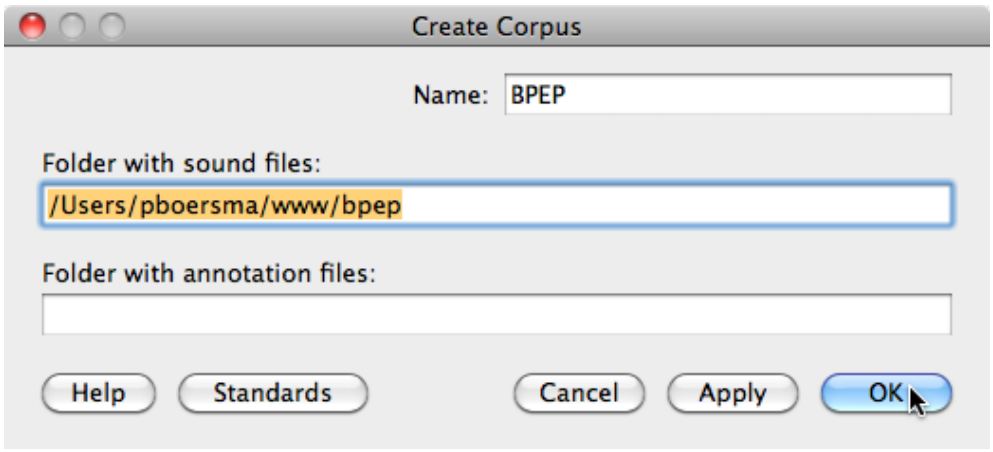
3.3 What else can be annotated?

Beside Sound objects, other types of objects that can be annotated are EEG (an electroencephalogram, which can be annotated for stimulus events) and Movie (a sequence of video frames).

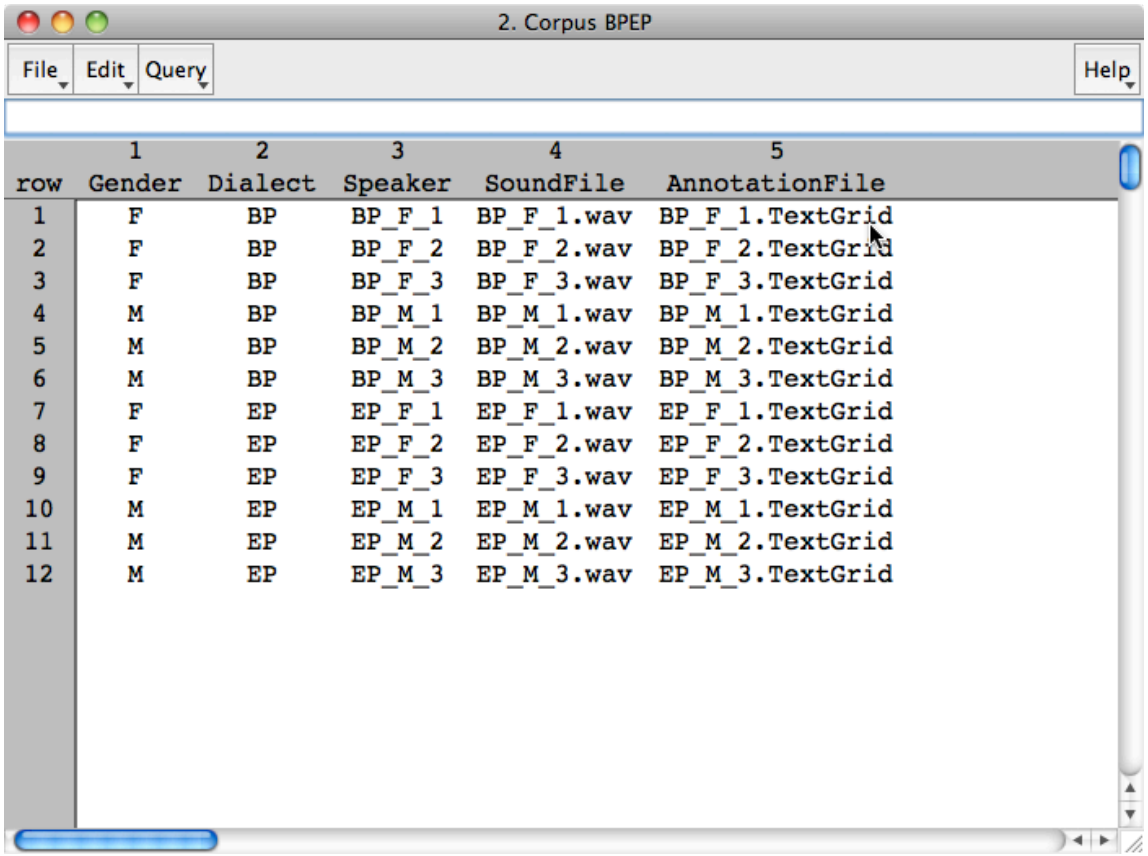
4. The Corpus window

A corpus usually consists of multiple sound and annotation files. Praat can collect these into a single Corpus object, as long as the name of each annotation file is identical (except for the extension) to a name of a sound file.

With **Create Corpus...** from the New menu you can specify a directory of sound files and a directory of annotation files:



In this example, no directory of annotation files is specified, so that Praat will look for the annotation files in the same directory as where it looks for the sound files. After you click **OK**, a new Corpus object appears in the list, and when you then choose **View & Edit**, a Corpus window pops up:



row	1 Gender	2 Dialect	3 Speaker	4 SoundFile	5 AnnotationFile
1	F	BP	BP_F_1	BP_F_1.wav	BP_F_1.TextGrid
2	F	BP	BP_F_2	BP_F_2.wav	BP_F_2.TextGrid
3	F	BP	BP_F_3	BP_F_3.wav	BP_F_3.TextGrid
4	M	BP	BP_M_1	BP_M_1.wav	BP_M_1.TextGrid
5	M	BP	BP_M_2	BP_M_2.wav	BP_M_2.TextGrid
6	M	BP	BP_M_3	BP_M_3.wav	BP_M_3.TextGrid
7	F	EP	EP_F_1	EP_F_1.wav	EP_F_1.TextGrid
8	F	EP	EP_F_2	EP_F_2.wav	EP_F_2.TextGrid
9	F	EP	EP_F_3	EP_F_3.wav	EP_F_3.TextGrid
10	M	EP	EP_M_1	EP_M_1.wav	EP_M_1.TextGrid
11	M	EP	EP_M_2	EP_M_2.wav	EP_M_2.TextGrid
12	M	EP	EP_M_3	EP_M_3.wav	EP_M_3.TextGrid

In this example, I have already extracted the between-speaker factors (Gender and Dialect), as well as the speaker names, from the file names; such information will be saved when you save the Corpus object to disk. When you later open the same Corpus file in Praat and the sound and annotation files have not moved with respect to the Corpus file, the information shown in the figure will still be available. The corpus object saves the locations of the sound and annotation files as relative to its own location; this allows you to move and distribute the Corpus file with the sound and annotation files without breaking the links between them.

The Corpus object allows fast annotation of a new corpus. If you click on the name of an annotation file, Praat will pop up this annotation, together with the corresponding sound, in a new TextGrid window (it is easy to **Save** this TextGrid to its original location on disk). If there is a sound file without a corresponding annotation file, the column AnnotationFile contains a question mark in the relevant row. When you click this question mark, Praat creates a new TextGrid and opens it in a new TextGrid window, together with the sound (again, the **Save** command will know where this TextGrid has to be saved).

The Corpus window allows querying all its TextGrid objects at the same time. You can search for entire texts, substrings, regular expressions, and so on. The results will appear at the bottom of the Corpus window, and a click in that window will take you to the corresponding location in an annotation and waveform.

Automated acoustic analyses of e.g. pitch and formant values of the whole corpus have not been implemented, because the number of possible research questions is simply too high. You can implement such analyses yourself by programming them in the Praat scripting language, as described by Caren Brinckmann in an accompanying chapter.

References

Bird, Steven, and Liberman, Mark (2001). A formal framework for linguistic annotation. *Speech Communication* **33**: 23–60.

Ladefoged, Peter (2005). *Vowels and consonants: an introduction to the sounds of languages*. Second edition. Malden, Mass. & Oxford: Blackwell.

Ladefoged, Peter, & Ian Maddieson (1996). *The sounds of the world's languages*. Oxford: Blackwell.