Software DIAOULEK 1.4

User's book

Table of contents

Introduction

Software “ Diaoulek ” is a software to help you learn languages, any language, but as this is a new software, it does not have already made lessons, except for the couple : language to be learned = Breton, language of reference = French. However a dozen of English-French lessons are also available.

In software “ Diaoulek ”, the vocabulary to be learned is divided into small lessons, each of them having about twenty entries (words, expressions, sayings...).The exact number of entries is not limited in one given lesson. We can, also, have as many lessons as we want. The set of all available lessons is making the data-base of “ Diaoulek ”. All the words (more exactly : entries) of the data-base can be reused into your personal lessons. You will then take advantage of the audio files that may come together with the published lessons. Your own lessons will complete the data-base. You can also generate “Prov” (provisional) lessons. These lessons have a limited life span, they are composed exclusively of pointers towards entries in the “regular” lessons. They are generated by simple clicks on buttons and they can be very useful for the revision of selected words during a limited time span. You can also create lessons said “Own” lessons in the same manner than the “Prov” lessons are created. However the life span of the “Own” lessons is infinite and they can be modified by the user.

With software “ Diaoulek ”, you can study different languages, each language will have its own configuration file. Moving from one configuration file to the next is done by the the order “ !chconf ”.

When you are studying a language, it is necessary to learn the vocabulary in the two directions : from the language to be learned to the reference language and from the reference language to the language to be learned. By convention, software “ Diaoulek ” is naming “ Question ” the text written into the language to be learned and “ Response ” the text written into the reference language. Thus, it is necessary to study the language in both directions : Question → Response and Response → Question. For software “ Diaoulek ”, both directions are independent but you can change direction very easily by a simple click of the mouse on the appropriate button.

Software “ Diaoulek ” is registering (after confirmation) your results, that is to say what you have declared for each word when you were learning the lesson. For each entry in the data-base, the software registers the number of interrogations with a positive response, the number of interrogations with a negative response, the date of the last interrogation with a positive response and the date of the last interrogation with a negative response. Software “ Diaoulek ” tries also to determine for each word the maximum length of time you can allow between two revisions. As both directions Q → R and R → Q are independents, we have a total of 10 values recorded for each entry.

With its registered values, software “ Diaoulek ” is able to estimate how well you know each vocabulary entry that was studied. For each entry, it will allocate two errors (one on each direction). The software can also compute for you heterogeneous lessons, that is to say lessons whose entries are taken into various lessons, for example the most badly known entries in the data-base, vocabulary studied in one direction and not the other, etc...

It is also possible to attribute to each entry one or more tags, this will allow you to selectively study the vocabulary of the land, of the farm, the bird or tree names, etc... For that, it is enough to ask software “ Diaoulek ” to compute an heterogeneous lesson with entries having all of them the same specific tag.

The separation of the vocabulary into short lessons is well suited to the learning phase but heterogeneous lessons (many kinds of them exist), are more suited to later reviewing.

Even if learning vocabulary is the software's main objective, “ Diaoulek ” may also display what is called complete lessons, like in a book, but a book the would be able to speak. This allows you to hear and hear again the lesson's sentences. The study of the vocabulary can also be made an integral part of such lessons. As the software is new, no method is using this possibility yet (at the end of 2012). However, you will find some 10 complete Breton → French lessons that were written to devise and tune the software. They are published with no guaranty at all ! Send me your corrections...

Software “ Diaoulek ” is written in C and should work on Linux and MS-Windows. It is only making use of the graphic library GTK and of the audio library “ libsndfile ”.

What is new on version 1.4

On version 1.4 of software “Diaoulek” the following items have been introduced :

1. A trash can is available for the “Prov” lessons. You can recuperate your “Prov” lessons during 5 days after their official removal.
2. In order to increase life-span of USB keys, all provisional files have been transferred to the temporary directory (/tmp).
3. A compact display of the vocabulary lessons is proposed now. Though this mode has some limitations, it also let you spare a lot of time.
4. The software uses now customized check boxes. They are larger and more easily readable.
5. A reverse dictionary is now available. You can select words in the language of reference to make “Prov” lessons (by the order !shcid).
6. The updating of the Breton lessons is now very simple by the order “!update”
7. The “Own” lessons have been introduced. They are very much like the “Prov” lessons but their life span is infinite and you can modify them.

What was new on version 1.3

On version 1.3 of software “Diaoulek” the following items have been introduced :

1. When the Question is displayed, a sound button has been introduced. This button allows you to hear the question again, even if the sound is off.
2. When leaving a lesson, you can create a “Prov” lesson. It is a lesson with a limited life span. It is useful to review the vocabulary.
3. The software makes a dictionary of all the entries in your data-base. You can select entries to make a “Prov” lesson. You can also make dictionaries of entries with a given tag.
4. You can add and study “Prov” lessons.
5. You can study different languages on the same session of software “Diaoulek”. For that, you simply have to shift to another configuration file.
6. New computation of the error associated with the studied entries. These errors should now be equal to the probability that the word or expression is unknown. Introduction of the

notion of limit date. After this date, the probability that the entry is unknown should be equal to 100%. Full recording of the 10 parameters associated with an active entry in the LOG directory.

7. Improvement of the order “import” which is used to make a copy of the lessons and recorded data when you change your version of software “Diaoulek”.
8. Improvement of the orders “!synchro” and “!synchro0”.
9. Software “Diaoulek” can now be internationalized. You just have to complete file ”diaou_internat.txt”.

How the software looks like.

General Presentation.

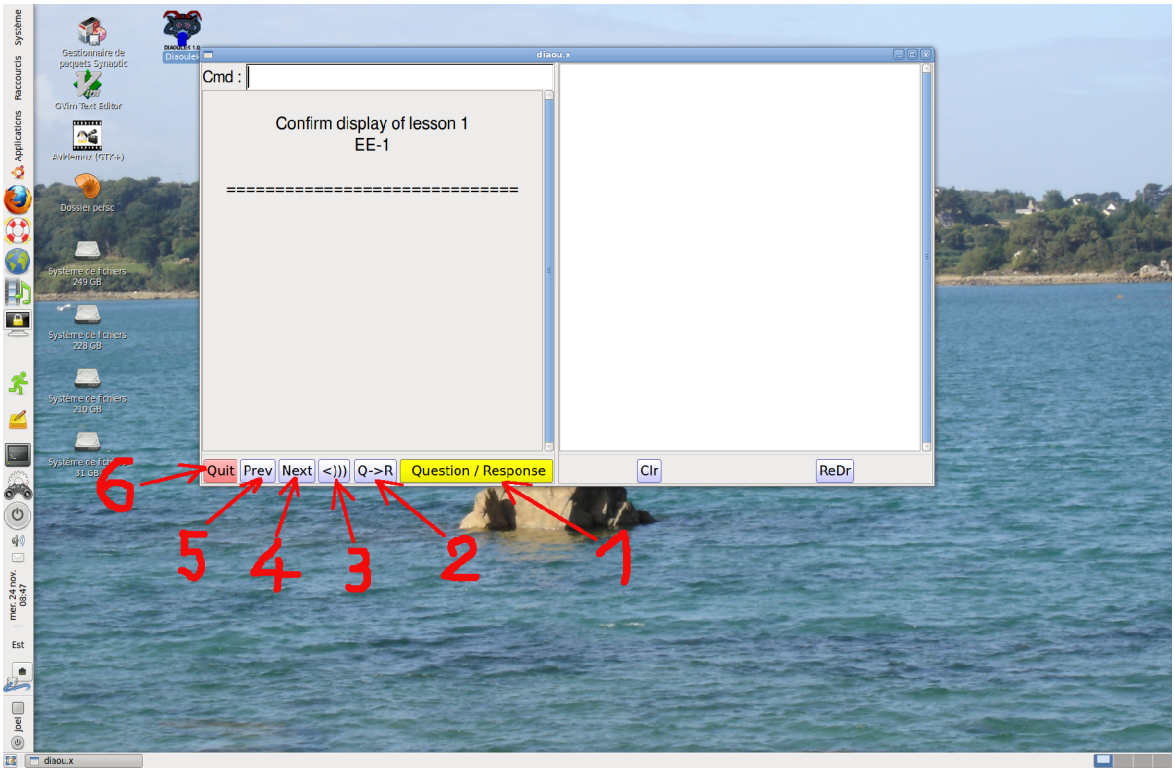


Figure 1 (Diaoulek-0.png) *Whole screen capture showing the different buttons.*

The main window of software “ Diaoulek ” is divided in two parts : Left part is the "vocabulary window" where you will learn or review the vocabulary. The right part is the “lesson window” where the few complete lessons will be displayed. This “lesson window” will also be used to display dictionaries of the all the entries in your data-base as well as many other things such as the list of all your “Prov” lessons.

The button labeled 1 is the main button of “ Diaoulek ”. If you can master the use of this button, you will have understood almost everything in software “ Diaoulek ”.

This button has two states : the yellow state and the red state. When in the yellow state, it is simply waiting for a mouse click to make the software continue. In the red state, it is ready

to register the meaning of your next click and make the software pass to the next step.

This button is able to discriminate between the different clicks generated by the mouse, the left click (the usual one) means : yes, it is good, OK... The right click means : no, it is not good, KO... The middle click means : I pass, forget the previous recording, let us continue without any recording...

All the other buttons are good and honest buttons which are doing only one single thing when pressed.

Here is on the following table the various actions of the main button according to its state :

State :	*Left click*	*Middle click*	*Right click*
Yellow state	Continue	**Erase the recording of the previous word**	Continue
Red state	Record as good and continue	Pass the recording of the present word and continue	Record as false and continue

Remark 1 : The records (“ good ” or “ false ” results) are only provisional, they will become definitive only when leaving the lesson and after a demand of confirmation by the software.

In fact, the main button is into its yellow state during the first part of the interrogation and then it toggles to the red state when we are in the second part which is the verification phase. The red means that the software is ready to register your response. The middle click erases the recording of the previous word in the yellow state but it does not make the program to continue, the software is still waiting for a left or right click. In the red state, the middle click don't register anything for the present word but we continue.

Remark 2 : in the interrogation phase (yellow state of the main button), the left or right clicks of the mouse have the same effect, to make the program continue, but it is better to make a left click if you think that you know the meaning or the translation of the word and you will do a right click in the opposite case if you are sure that you don't know. If you do that, during the next phase, the verification one where the recording takes place, the click will become an almost instinctive reaction and you will have fewer errors.

As it has already been said, by definition, software “ Diaoulek ” is calling "Question" the words or expressions in the language to be learned and "Response" their translation into the reference language. One must learn the vocabulary in both directions : Question -> Response and Response -> Question. The button labeled 2 allows you to toggle between these two directions Q->R and R->Q.

The learning of a living language without audio records is almost impossible. The default behavior of “ Diaoulek ” is to always search for audio files. However, it is sometimes better to cut off sound, for example to accelerate the learning of a lesson that was already studied. The button labeled as 3 allows you to toggle easily from one state to the other. The " <))) " logo symbolize a loudspeaker in activity and the logo " <XX " symbolize (even more badly) a blocked loudspeaker.

My advice is to divide the learning of a language into small lessons, each with some twenty words into them. Button 4 (Next), allows you to go to the next lesson and button 5 (Prev,

put here for “ Previous ”) allows you to go back to the previous lesson. There is no button to end the learning of a lesson, the “Next”, “Prev”, “Quit” buttons or the use of the command line to ask for some action will terminate the present lesson.

The button labeled as 6 (Quit) makes you leave the application while registering the present state (last studied lesson, position and size of the windows...). You will be back to that state next time you use “ Diaoulek ”.

Remark 3 : You can study any language with “ Diaoulek ”. However, as the software is very young, it is only for the pair Breton → French that about 200 ready-made lessons are available at present. You will be able to download 10 full lessons. Though they were mainly written for technical reasons, the code's adjustment, they can also be used as an introduction to the study of Breton language. Together with the other vocabulary lessons they constitute a data-base of about 5000 words or expressions (June 2012) which can be used into your own lessons. All the lessons, the full ones and those of the data-base are given with their audio files.

Remark 4 : Figure 1 was taken on a computer with an Ubuntu 10.04 operating system, with a MS-Windows system the color of the buttons may be slightly different as the versions of the graphic library GTK are also different. The Windows versions Diaoulek1-3 and 1-4 have been generated on a Linux Mint 13 with Mingw and then tested with ”Wine” on Linux and finally verified on Windows XP and Win7.

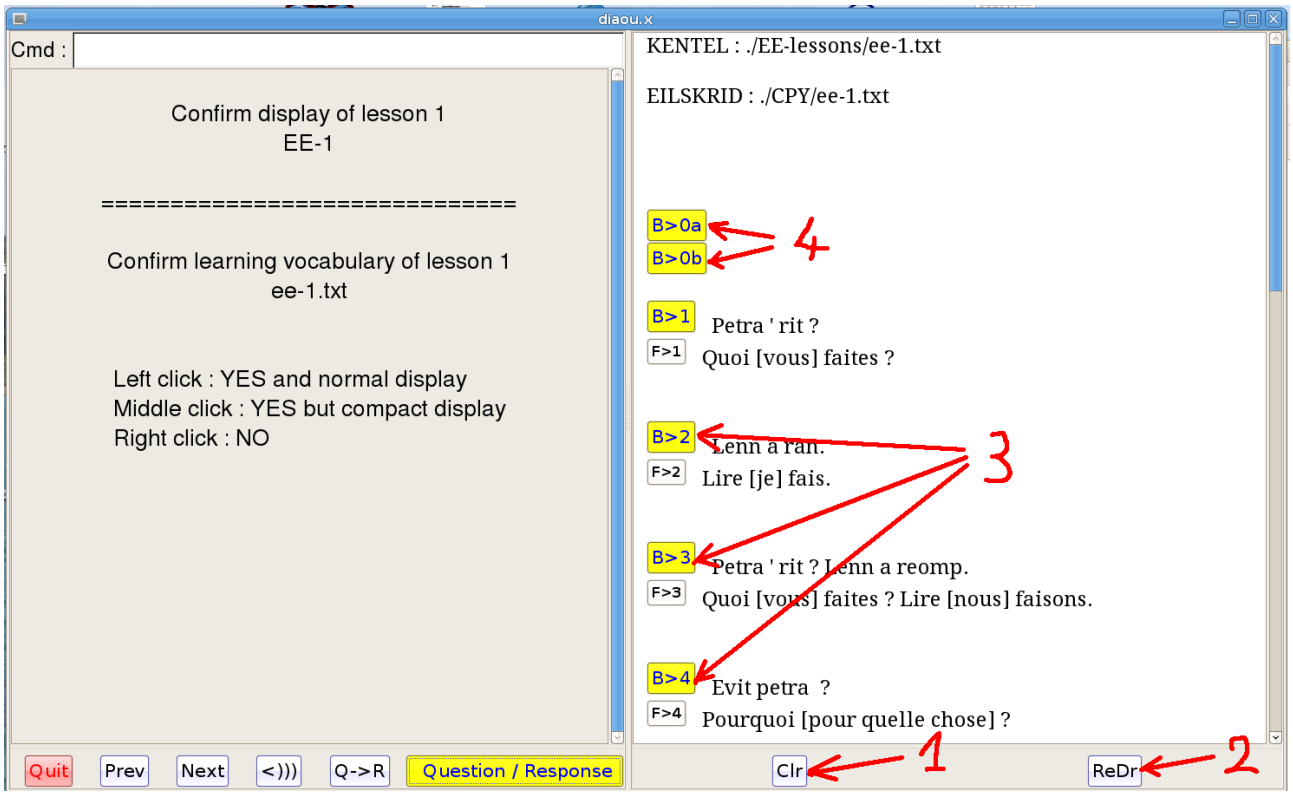


Figure 2 (Diaoulek-1.png) *Display of a full lesson.*

Even if the main objective of software “ Diaoulek ” is the learning and review of the vocabulary, a secondary objective is the display of classical lessons, like in a book, but a book that is able to speak.

The button with label 1 (Clr pour "Clear") allows you to erase the lesson window, this can be useful when you are learning the vocabulary of that lesson.

The button with label 2 (ReDr pour "ReDraw") reestablishes the contents of the lesson window.

The buttons with label 3 in the lesson windows are audio tags that allow you to hear and hear again each of the sentences of the lesson.

The buttons with label 4 allow you to hear and hear again the full lesson. That is however specific to that lesson and may not exist on other lessons.

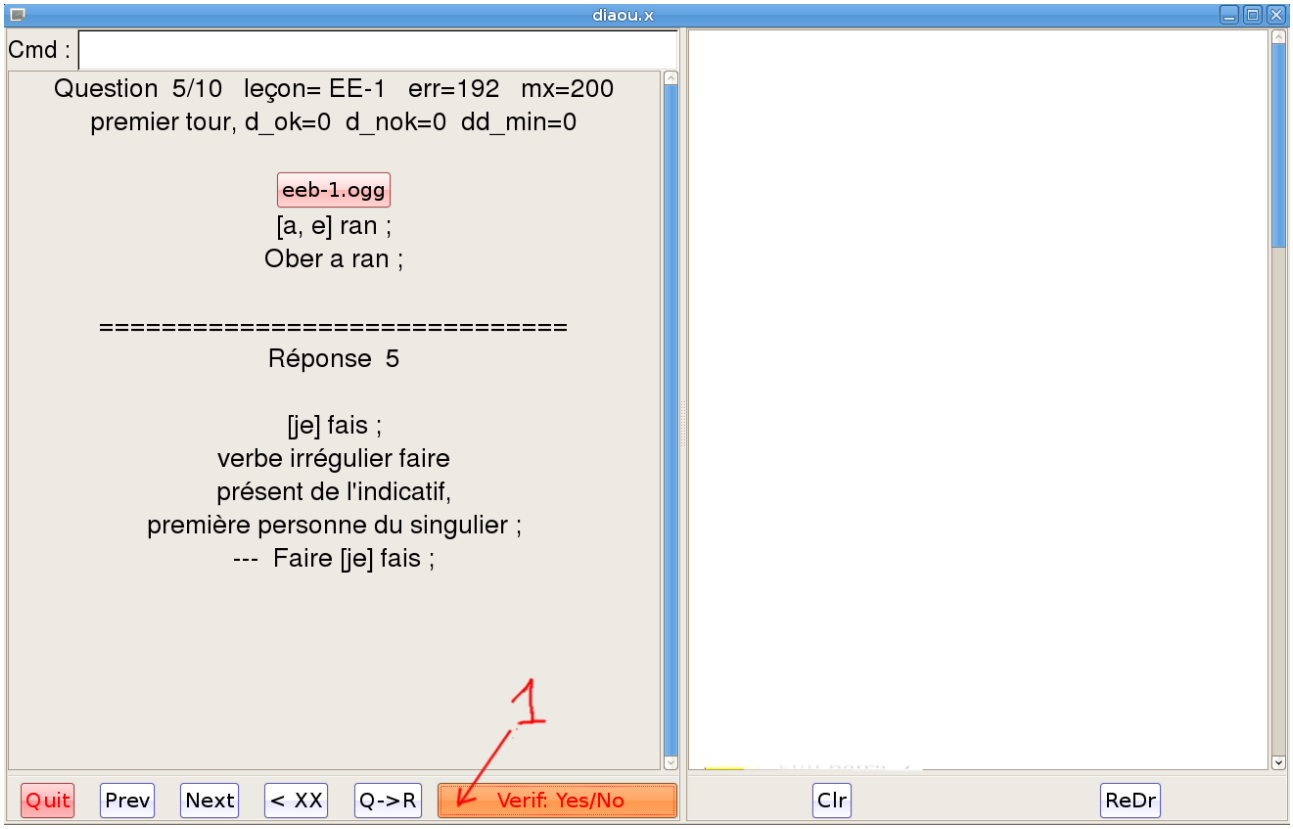


Figure 3 (Diaoulek-5.1.png) *Learning of a lesson's vocabulary.*

We are now for this full lesson in a phase of vocabulary learning. The lesson window has been erased, the sound is off as indicated by the button " <XX ". The learning is done in the direction Question → Response as indicated by the button " Q->R ".

As the sound is off, it is a writing interrogation but you can still have the sound by the clicking on the red button labeled "eeb-1.ogg" (this is the name of the sound file). When the sound is active, the interrogation is an oral one, at least when an audio file is available. The Breton writings would have then appear only in the verification phase, that is to say under the line "==========".

On figure 3, we are already in the verification phase and the main button (labeled 1) is in its red state. It is ready to register your click. A left click to say "it is good", a right click to say "I don't know", or a middle click to say "jump, forget that question, and don't record"

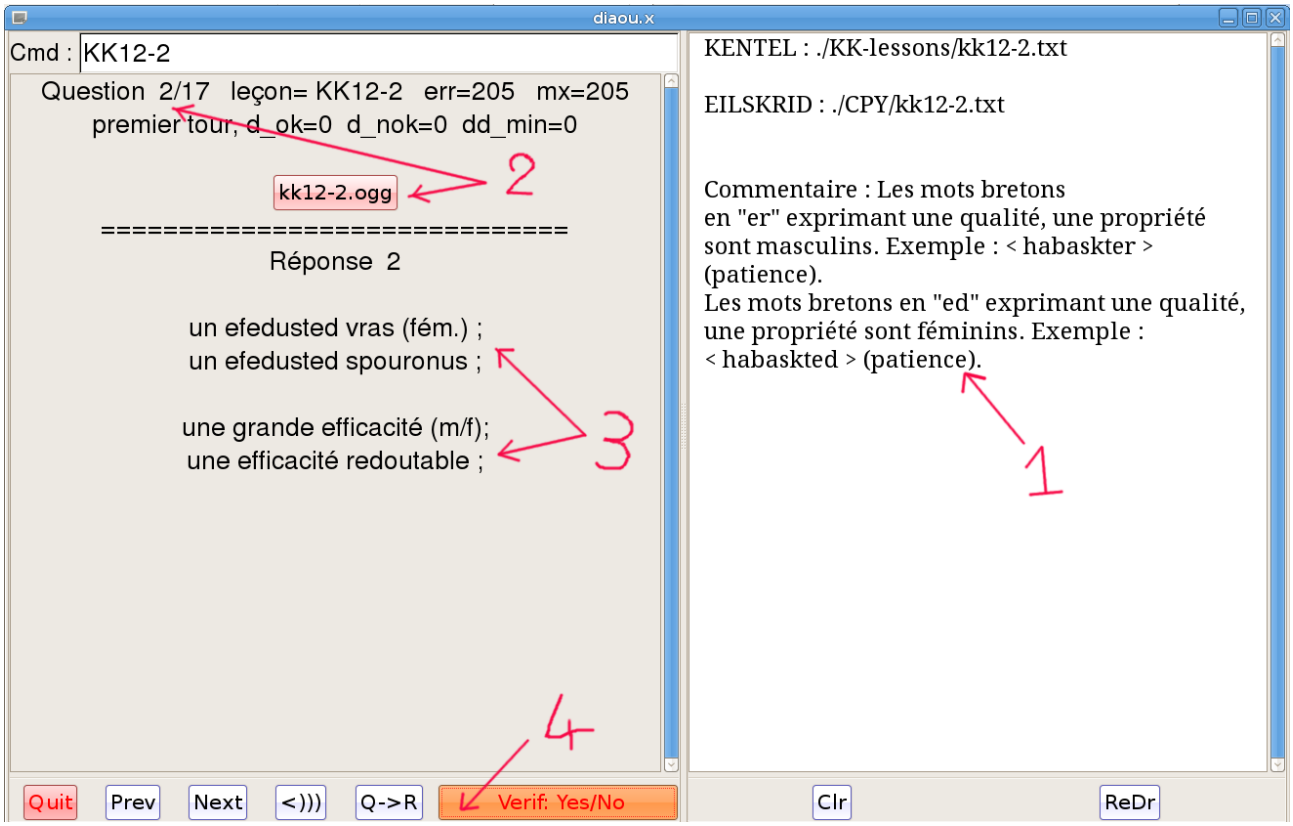


Figure 4 (Diaoulek-6-1.png) *Display of an almost ordinary lesson.*

Even if we have only very few full lessons, some lessons can still include useful comments like here on the window labeled as 1.

In the case of figure 4 the sound is active as indicated by the button "<)))". In the interrogation phase, before the line "====", we find, as indicated in 2, the number of the question (here 2 on a total number of 17), the name of the lesson (here : KK12-2) and the name of the sound file (here : kk12-2.ogg) written as the label of a button that you may click in order to hear again the sound. Most often several words are on the same sound file but this is taken into account by the software and only the right words will be heard. There are also other indications that will be explained on the following figure.

In the verification phase (labeled as 3), under the line "====", the words or expressions in Breton with their translations in French are given. It is then possible to verify that the Breton speech was well understood and that the translation is known... or the contrary.

As in the case of the previous figure, the main button (marked here as 4) is waiting for your click to make a provisional recording.

Version 1.4 of software “Diaoulek” let you choose between the normal display of a lesson as explained above and a more compact display as shown on figure 5 below. With a compact display, you don't have all the possibilities of the normal display but it will save you a lot of time simply because you will not verify all the words of you lesson. You will only check the words that you are unsure of. This is the case on figure 5 where the line with the yellow word “roued” was clicked for verification. You have then on the left part of the window a

display of the Breton word and of its translation in French.

The compact display mode for the lessons.

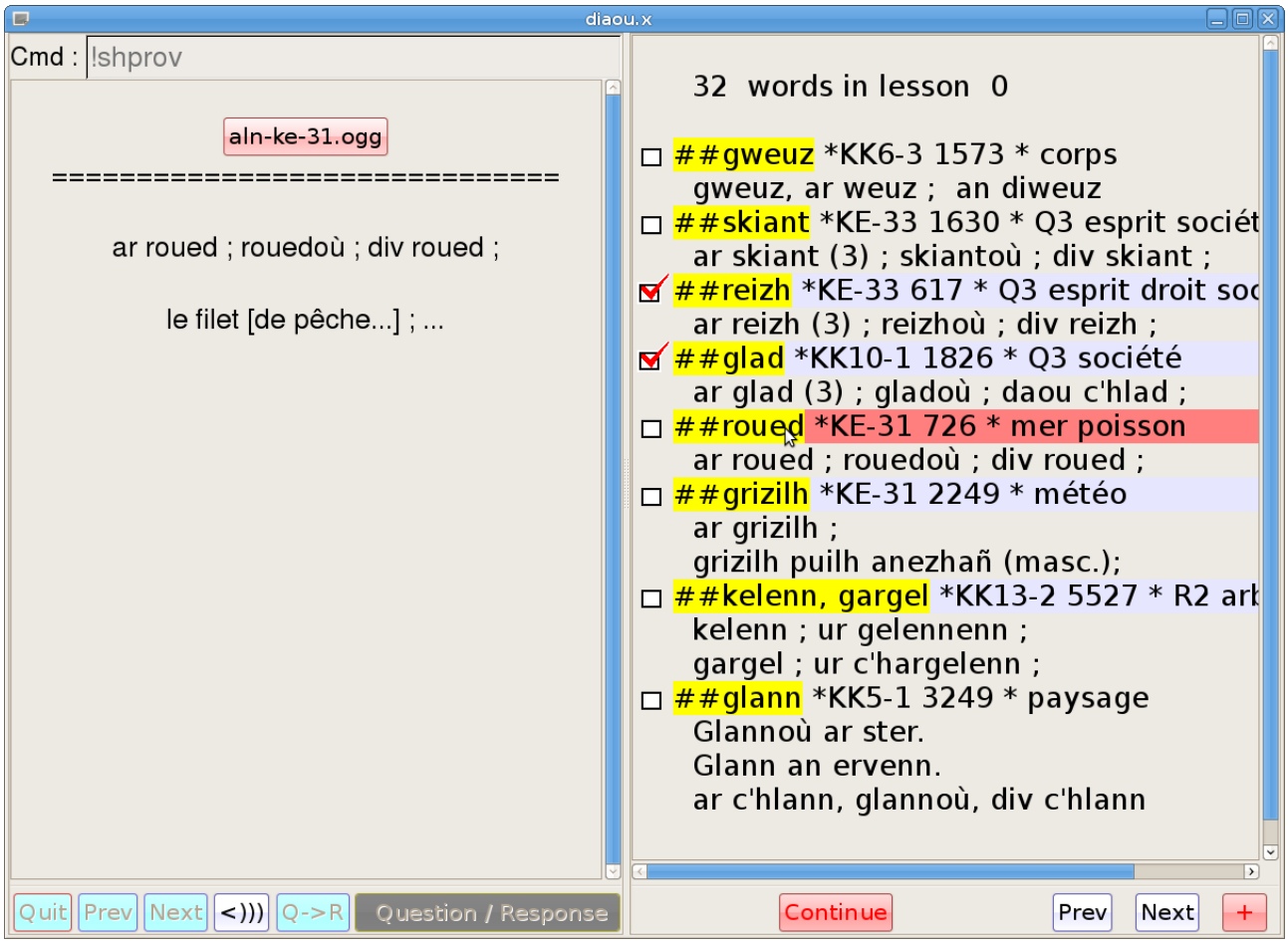


Figure 5 (Diaoulek-16-en.png) *Compact display of a lesson*.

On the right part of the window, the check boxes allow you to select some words, for example those unknown (the KO words). All the other words will then be considered as known (OK words). You can also do the opposite and check only the known words if they are less numerous than the unknown ones. As you may not have enough place on your screen to display all the words of your lesson, it will be divided into “pages” and you will change the displayed page by a click on the “Prev” or “Next “ buttons at the back of the window.

Once you have finished to learn your lesson, you will press the “Continue” red button. This will show you a dialogue window where you will be asked to activate a button in each of the three groups of buttons. These buttons will allow you to create a provisional lesson (Prov lesson) with the selected words and to register or not the study of your lesson. We will see later in details what a provisional lesson is.

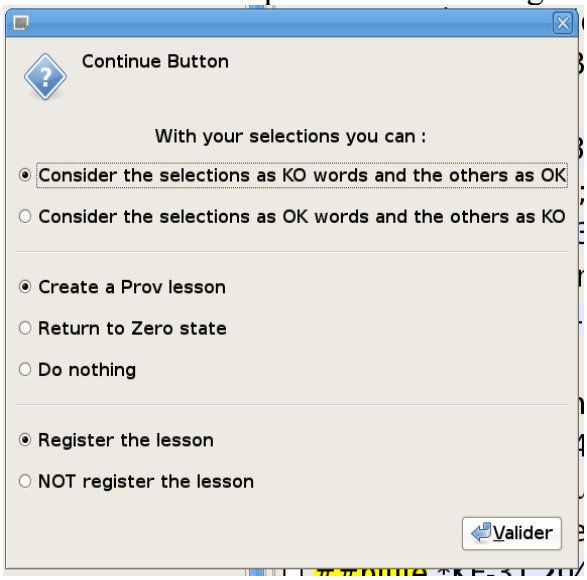
Remark 1 : When the lessons are displayed under the compact form, many buttons and the command line at the top of the vocabulary window are inactivated to avoid bad interferences. In fact, you can only activate or inactivate the sound. The other buttons and

the command line are reactivated when you leave the compressed display mode by pressing the continue button.

Remark 2 : The words on a compact display of the lessons are in random order. This order changes at each new study and the probability that you obtain twice the same order decreases very rapidly with the number of words in the lesson (as 1/n!).

Remark 3 : After a study with a compact display of the lesson, the same lesson but in the other sense (QR or RQ) will be proposed. You can agree or change the lesson by pressing the “Prev” or “Next” buttons or by using the command line at the top of the vocabulary window.

You can see below a picture of the dialog window.



On this dialog box, you have 3 groups of buttons. You must choose to activate one button on each group. On the first group, you have two possibilities, consider the selected words as KO and the unselected ones as OK or choose the opposite. On the second group of buttons, you can choose what to make with your selected words, create a provisional lesson (see later), return to the zero state (never studied word) or do nothing. The third group of buttons let you choose between registering or not the lesson.

Presentation of the command line :

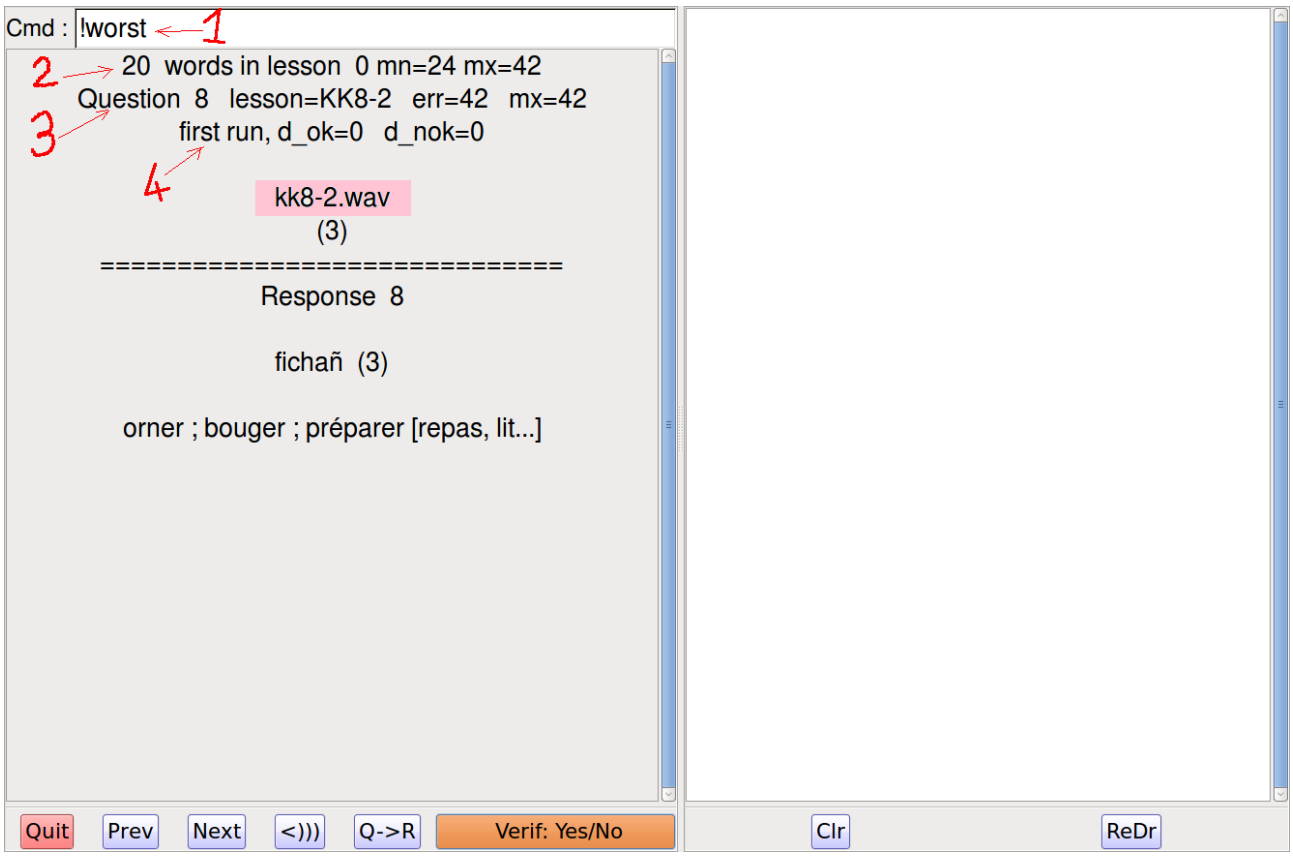


Figure 6 (Diaoulek-7-1.png) *Making use of the command line*.

The command line (labeled as 1), at the top of the vocabulary window, allows you to reach a given lesson by its number or by its alias without having to use the "Next" and "Prev" buttons. But a lot of other commands for different objectives are also at your disposal. All these commands are beginning by an exclamation mark.

It is possible to make the software compute for you composite lessons where words are chosen according to various criteria. For example a selection of the most badly known words, words that were not already studied, words with a given tag (vocabulary of the sea, of the farm, of the house, bird names...). Obviously we make the difference between the directions Q->R and R->Q. A list of all the available commands can be obtained by typing "help" (without the " ") in the line with label 1.

In the case of figure 6, we have, by the command "!worst", made software Diaoulek to compute a lesson with the 20 most badly known words in the present direction Q->R. These words are taken among those already studied. As the software is registering for each word and each direction the number of interrogations with a positive result , the number of interrogations with a negative result (badly known word...) but also the dates of the positive or negative interrogations, it is able to find the most badly known entries among those already studied. To do that, the software takes into account your previous results as well as a forgetfulness factor that it has tried to guess for each entry in the data-base. Computation time is negligible even on an old computer.

The software “Diaoulek” gives to each word an error. A large error indicates a badly known word and an error equal to 0, a well known word. If things where as hoped this error should be equal to the probability (scaled from 0 to 100) that the word is unknown. However, the error can be superior to 100. In that case, the difference "error -100" is the number of days

since the software consider that particular entry as completely forgotten. The software is rather pessimistic and often you will have the good surprise to discover that you know the meaning or the translation of a word that the software consider as forgotten. The contrary may also happen ! In any case, the software will adjust and hopefully improve its prediction for this particular entry.

In the interrogation phase (over the line "===="), various informations are given. The line with label 2 indicates that there are 20 words in lesson 0. It is the usual number and all the computed lessons are said to be "lesson 0". But line 2 also indicates that in the present lesson the smallest error is **mn**=24 and the largest is **mx**=42.

On the line with number 3, it is also indicated that we are displaying question 8 which is sending you to lesson KK8-2. This word, in the present direction Q->R does have an error "**err**=42" and the maximal error in lesson is also "**mx**=42". We then are dealing with the most badly known word or at least one of a group of them if several have the same error. Software “ Diaoulek ” makes its interrogations in random order but it will insist on the badly known words. It does have some inclination to be slightly "diabolic".

The game's objective is to decrease **mx** towards zero. At that point the lesson can be considered as having been learned.

The lesson's learning is made in two steps. During the first step (as indicated here by "**first run**" on the line with tag 4), we will see successively all the words in random order. When all the words have been seen at least once, we pass to the "**second run**" during which the software will strongly bias it random order towards the most badly known words.

On the line with tag 4, "**d_ok**" and "**d_nok**" are the numbers of good and bad results for the learning of that word today in the present lesson (it is not the whole history record of that word). When we are in "**first run**" these numbers are both equal to zero. During the "**second run**", at least one of these numbers is not equal to zero.

If we don't have enough time, we will stop at the end of "**first run**". When we will enter into the “ **second run** ”, we will be sure that every word was seen at least once. If we have a little more time, we will continue to make "**mx**" decrease and reach a zero value. At that point, the lesson will be considered as known, at least for the day...

Creation and use of a provisional lesson.

As indicated earlier, there is no special button or order to terminate a lesson. For that, you simply need to take another lesson, write any order in the command line or leave the application. However, the results of your study have not been permanently registered yet. For that, the software will ask for your authorization and you have also the possibility to make other actions by a selection of your lesson's entries. One such action, the creation of a provisional lesson will be detailed here.

Creation of a provisional lesson.

You have just studied a particular lesson and there was for example 5 words that were difficult or that you did not know. It would be nice to review only these 5 words tomorrow or during a few days up to the point that they would be well known. With only a few clicks you can do that ! With version 1.4 of software “ Diaoulek ”, you can create provisional lessons, that is to say lessons with a limited life span that contain only pointers towards words of the data-base. You can then list all your provisional lessons, select some of them and study the sum of your selected provisional lessons. After a few days (you choose that number of days !) a provisional lesson disappears but this is not a problem because after each study you can create another provisional lesson with the badly known words. The same word can be in more than one single provisional lesson but the software is smart enough not to be in trouble when you study a sum of provisional lessons.

It has been said above that the provisional lessons disappear when their life span has been exhausted. This is not totally true. In fact they can be recovered for 5 more days after their official disappearance. As a consequence, if you come back after 2 months of vacation, almost all your “Prov” lessons will be obsolete and removed but you can still recover them during 5 more days. Their name is only changed and will begin by “old”. These old “Prov” lessons are listed after the other ones.

A provisional lesson may be created on different occasions we will see here only one of them, when you terminate the study of a lesson. In that case, the software will show you something like the following picture :

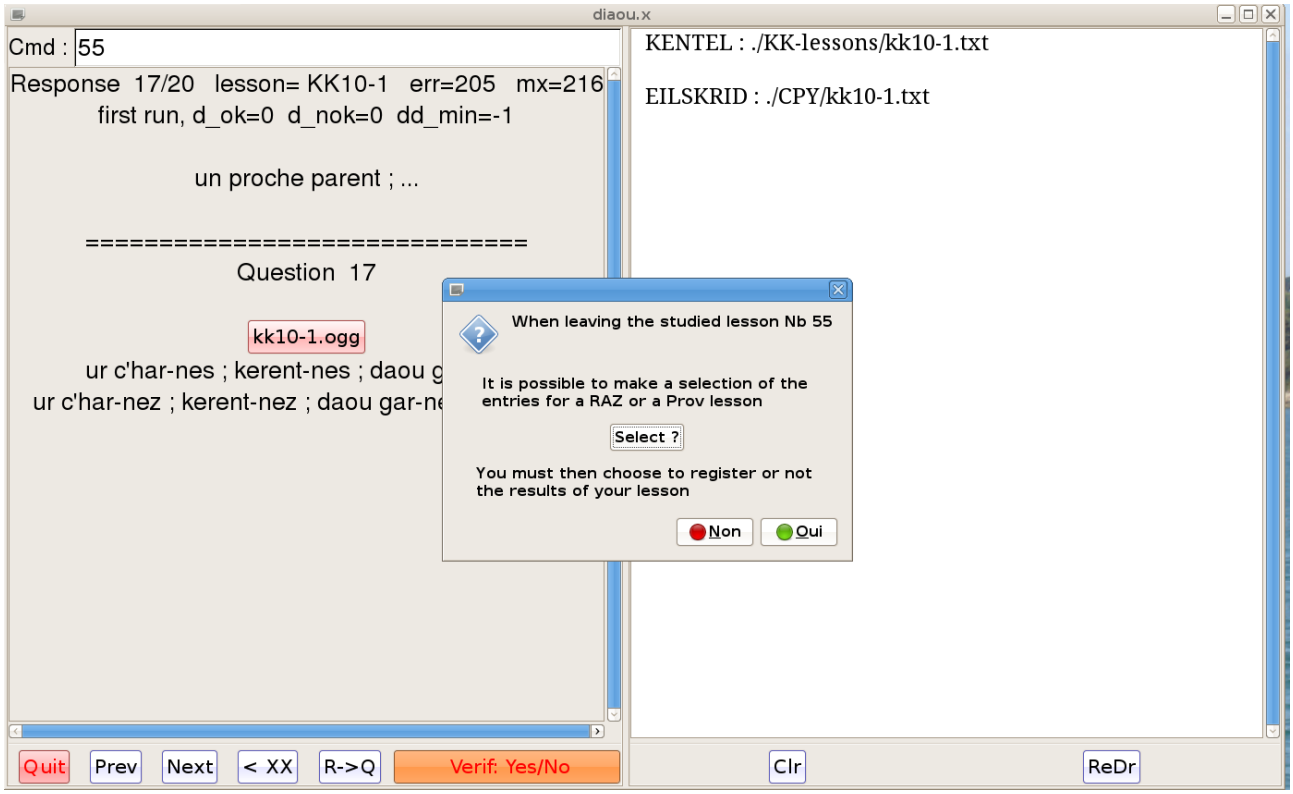
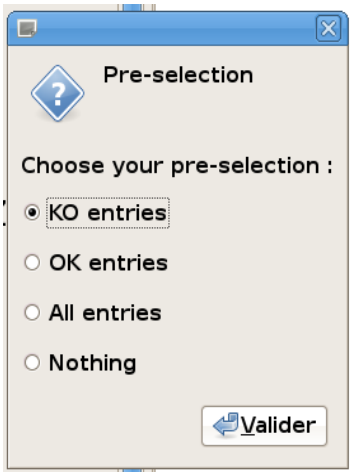


Figure 7 (Diaoulek-8-en.png) *When leaving a lesson*.

Before you choose to register or not your result, the software allows you to select some of your entries for a provisional lesson or even to put them back to the zero state (not learned state). You can do that by a click on the “select ?” button. You have then the following dialog box :



On this dialog box, you can choose to select your entries with a negative result (KO entries), or with a positive result (OK entries). You can also choose to select all the entries in your lesson or select nothing.

After validation, you obtain :

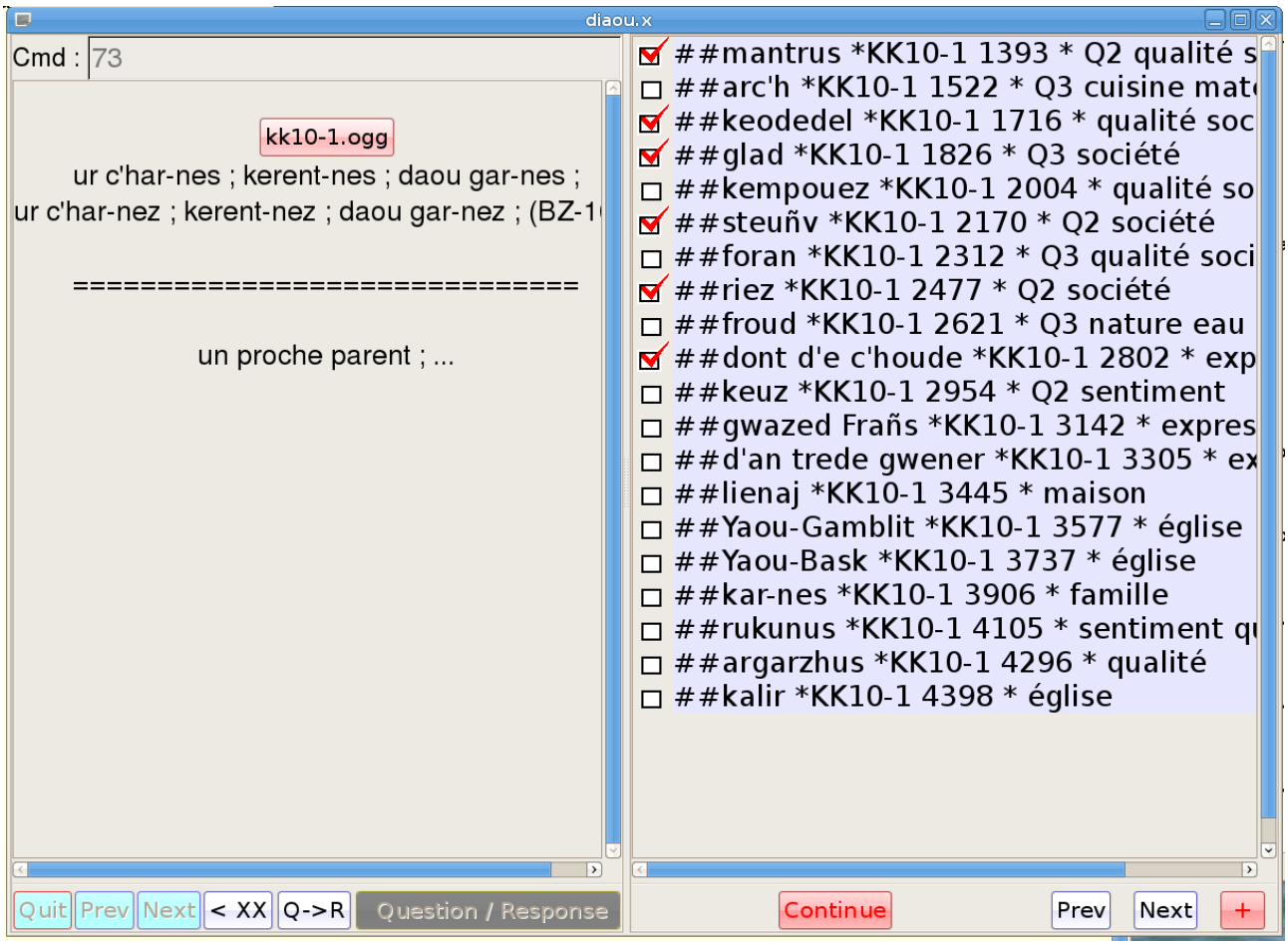
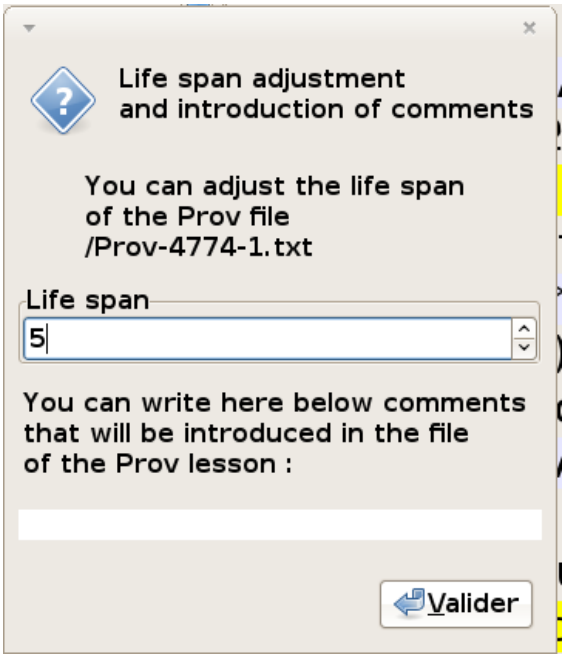


Figure 8 (Diaoulek-10.png) *Selection of entries in a lesson*.

The check boxes are allowing you to select entries in your lesson. Some of these entries are already selected, that was the purpose of the previous dialog window. You can select other entries or de-select some. Once you have done your choice, you can press the “Continue” button. Then a new dialog window will offer to you the possibility to create a “Prov” lesson or still to bring back these entries to the zero state (never studied state), or still to do nothing.

The creation of a “Prov” lesson is the default choice. After validation, you will have the following dialog box :



In that box, the software will give you the name of your "Prov" lesson. Here the name of the lesson is Prov-4774-1.txt, it was created on the 4273rd day after the first of January of the year 2000 and it is the first lesson created that day. You can also adjust the life span of your lesson (the default is 5 days) and you can add comments to your lesson. You can chose any positive number for the life span of your lesson. After that number of days, the lesson will disappear from the list of "Prov" lessons but will still be recoverable for 5 more days.

.

Once you have pressed the validate button, you return to a state corresponding to what is shown on Figure 7 because you have not made your choice to register or not your results. Again it is proposed to you to select entries in your lesson. Obviously you will ignore that choice, excepted on the rare case where you want to make another action, for example to return some entries to the zero state (never studied state). After a click on the "yes" or "no" button, the software will resume its normal course according to your choice, the learning of another lesson, the change of studied language, the leaving of the application...

Use of provisional lessons.

As we have seen, the creation of a "Prov" lesson is very easy. However, we need now to learn how to use that "Prov" lessons.

I am not a "command line fanatic" but command line is an unavoidable choice when you are short on time for code development. That is why the access to your "Prov" lessons is made by writing the order "!shprov" (**sh**ow **prov**) on the command line. You will then obtain something which will looks like what is shown on figure 9 :

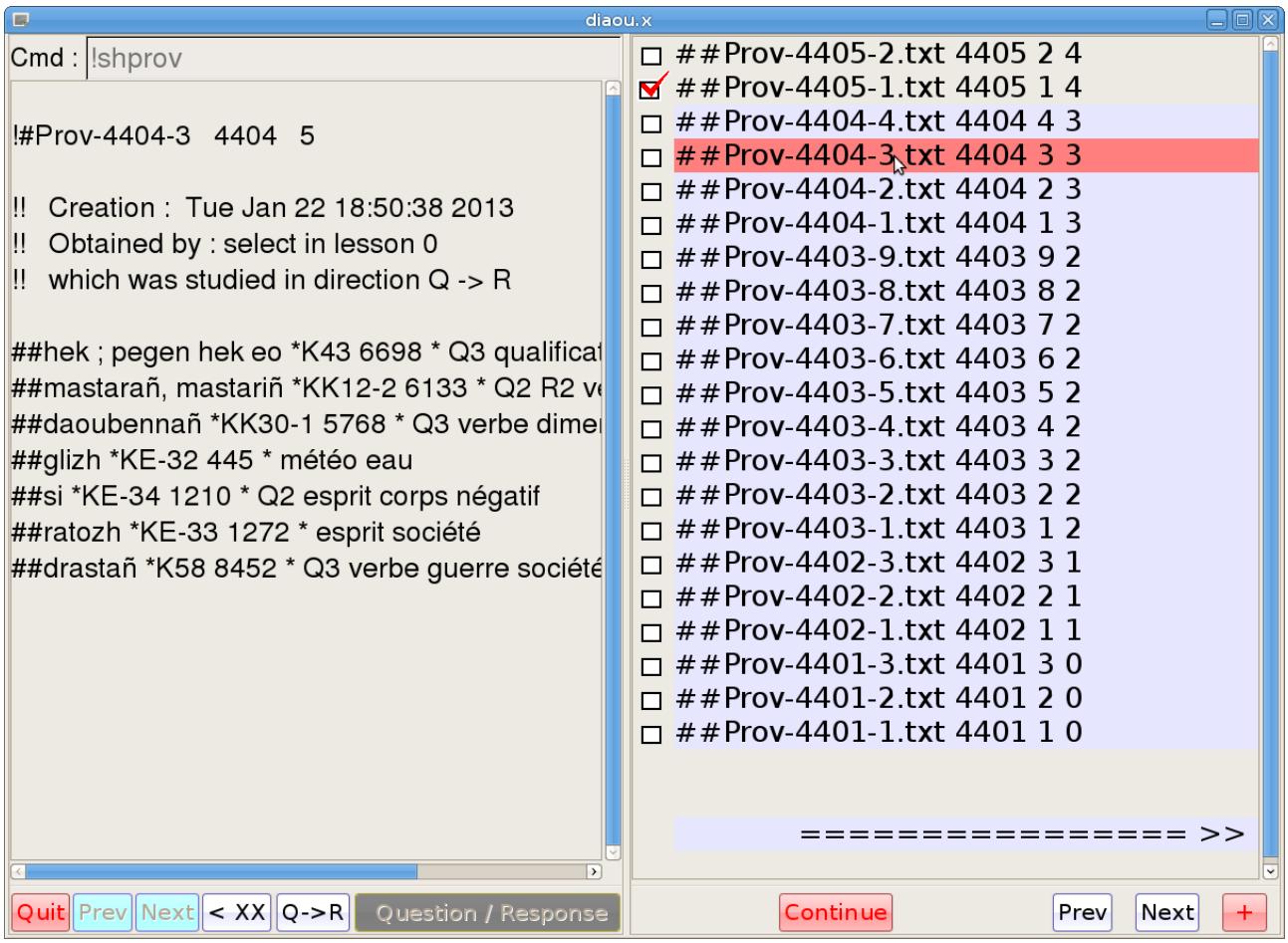
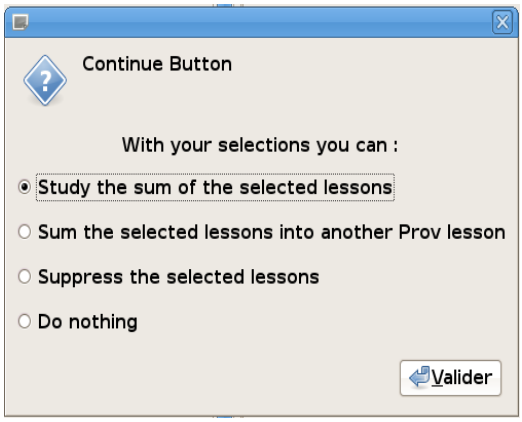


Figure 9 (Diaoulek-12.png) *Selection and display of the "Prov" lessons*.

As you can see on figure 9, on the right part of the main window (the lesson window), we have the list of all the provisional lessons. These lessons are classed by order of decreasing length of remaining life span. In fact the name of a lesson is a button that will turn to red when the mouse pointer is on it and, when clicked, it will display on the vocabulary window (the left part of the main window) an abstract of the lesson contents. You have the date of creation of the lesson and from where it comes. Here, on figure 9, the lesson was created on January 22, 2013, it comes from a selection of words on a lesson 0 (a compound lesson) that was studied in the direction Q->R. You have also the list of all the data-base entries reachable by this provisional lesson.

As you can also see on the lesson window, check boxes are placed before each "Prov" lesson name. They allow you to select some of these lessons, for example, all the lessons created while studying on the Q->R direction. After you have made your selection, you will click on the "Continue" button. That will open the following dialog box :



That dialog box will offer you some choices, the default is the study of the sum of all the selected lessons. If an entry is repeated on different lessons, that entry will be considered only once by the software. The sum of your selected “Prov” lessons is a kind of virtual lesson which is not different from the other lessons. In particular, after the study of this virtual lesson, you will be able to select words for the creation of another “Prov” lesson. So, a word that you have difficulties to learn may remain in the “Prov” lessons as long as necessary.

The provisional lessons are very precious for the learning of the vocabulary. They are created by a selection of words in a lesson (a true lesson, a computed one or the sum of other provisional lessons) or by a selection of words in a dictionary of data-base entries. They have a limited life span which avoids the useless accumulation of words. Still a word that you have difficulties to learn may be propagated from one provisional lesson to the next if you choose to select it again and again. Some recent studies on marine snail neurons suggest that learning can be more effective when done at the beginning by a series of reviews separated by short time intervals and then followed by other reviews separated by much longer time intervals. The learning of an unknown language by a grown up person may be different, however (we are not marine snails !), but you will note that the use of the “Prov” lessons mixed with the learning of computed “worst” lessons can allow you to follow such a learning method. It will nevertheless be up to you to find the mixture of new, “Prov” and “worst” lessons best suited to your person.

The “Prov” lessons are made exclusively of pointers towards entries in existing lessons. The “Prov” lessons are not conceived to be directly modified, for that purpose, you will use instead the “Own” lessons. You can also create a new lesson of your own and declare it into the configuration file. We will see later how to write a lesson from scratch.

A great part of the usefulness of the “Prov” lessons is the fact that they have a limited life span. Their disappearance allows you to learn new vocabulary. However, there is at least one case where this disappearance is a problem. Let us suppose that you have lessons with 5 days long life span and you leave for a month long vacation. When you come back, all your lessons are obsolete and the software must remove them. Still, you may want to reuse these lessons. That is why a kind of trash can has been devised for the “Prov” lessons. When a “Prov” lessons is obsolete, it will not disappear. Instead, its name will be changed. The new name will be composed of a prefix “old-digits-” followed by the ancient name. In the prefix, “digits” is the number of days since the first of January of year 2000 at the date of removal and not at the date of obsolescence. These renamed “Prov” lessons will remain available for 5 more day before removal. So, you can recuperate your “Prov” lessons 5 days after you are back from vacation. The obsolete and renamed “Prov” lessons are listed after the other active “Prov” lessons but on a following page reachable by the “Next” button.

Creation and use of an “Own” lesson.

As we have seen into the the previous paragraph, the “Prov” lessons are very useful but they have also limitations. You cannot modify them and at the end they disappear. The “Own” lessons are exactly like the “Prov” lessons but their life span is infinite and you can modify them with a text editor. They are created like the “Prov” lessons and they can be used by writing the order “!shown” (**sh**ow **Own**) into the command line at the top of the left window. At the creation of the “Own” lesson you will be allowed to change the file name and its alias, you will also be allowed to add comments like into the “Prov” lessons. You can add or remove articles into an “Own” lesson exactly as you can do into any regular lesson, that is to say in the simplified or more complete way. This will be explain into the next paragraph. The “Own” lessons are located into the “OWN” directory and you don't have to declare these lessons into the configuration file because the “OWN” directory is systematically scanned at the start of the session. You can also put these lessons elsewhere, into another directory, but then you will have to declare that directory into the configuration file.

Redaction of a simplified lesson.

At present, only a few complete lessons have been added to the release of software “ Diaoulek ”, they were written for technical reasons, code writing and tuning. Possibly, if nothing better is available, they can be used as an introduction to the study of the Breton language starting from French. It is however better to study a language with the help of a classical and popular method and limit the use of software "Diaoulek" to the learning of the vocabulary found in your method's lessons. For that purpose, you will write your own lessons, limiting them, it is an advice, to something like twenty words each. We will see now how to write a simplified lesson.

Composition of a simplified lesson.

Name of the file.

A lesson is a text file, you can choose any name for the file but it is better to make it simple and still end those names by the extension “ .txt ”, in order to show explicitly that we are dealing with simple text files. If you also intend to write more than one file of the same series, it may be good to use file names with a number. For example, you can name your personal files : perso1.txt, perso2.txt, … , perso10.txt. With that convention, you can, in the configuration file “ diaou.conf ” introduce the list of the lessons under the form perso1->10.txt and even if you want to begin by the last lesson : perso10->1.txt. That is much more easy when typing than the full set : perso10.txt perso9.txt … perso1.txt.

The name or alias of the lesson.

Each lesson must receive an alias (in fact a name). This alias must be composed of a few well chosen letters or numbers. For example in the case of the personal lessons that were mentioned above, the aliases P1, P2, … , P10 would be well suited. Another possibility could have been PP-1, …, PP-10, as the sign “ - ” is allowed. The aliases have to be short and easy to remember. They will be used in the command line of the vocabulary window to call the lessons. If you have 200 lessons, it is easier than having to press 199 times on the

“ Next ” button in order to reach the last lesson.

Composition of the lesson.

The lesson's first line must begin by the characters “ **!#** ” and then a space and the alias of the lesson. On the following lines, the couples questions and responses will be introduced by the signs “ **#** ”, “ **Q>** ” and “ **R>** ”. One example will be more explicit. Here is your first personal lesson :

```
!# P1
 !  Here is my first personal lesson
#
Q>   ti
R>   house

#
Q>   heol
R>   sun
```

The lines beginning by the sign “ **!** ” are commentaries and they are ignored with exception of the first one when the exclamation mark is followed by the sign “ **#** ”. In that case, the software looks for the lesson's alias.
We can put anything we want after the “ **Q>** ” and “ **R>** ” marks, even a few text lines. We only have to know that the lines will be displayed as they are and centered. So, you must not write lines that are too long.
A slightly more elaborate example is given in the file “ ex_simple_bis.txt ”.
In that lesson I am using some personal conventions which are useful only for the study of the Breton language. Here is one example :

```
#
Q>  ur gleizenn ; kleizennoù ; div gleizenn ;
R>  une cicatrice ; pluriel ; deux cicatrices ;
```

In my lessons, a Breton word is generally introduced by an article and this may induce a mutation, the word is followed by its plural (without any mutation) and the gender is indicated by “ div ” (pronounced “ diou ”, for feminine words) or “ daou ” (for masculine words). These two words “ div ” and “ daou ” which mean “two” for feminine and masculine nouns are inducing softening mutations. For the translation in French, everything can be translated like in the above example but most often the plural and the gender indication are replaced by three dots like in :

```
R>  une cicatrice ; …
```

Another of my personal conventions is the indication of the number of expected translations like below :

```
#
Q>  strafuilhañ (2)
R>  troubler, inquiéter
```

Obviously, you can use or not that kind of conventions or take some others of your own.

Reuse of entries belonging to other lessons.

It is very useful to be able to reuse the words or expressions belonging to other lessons. If these lessons have been published on Internet and moreover with their audio files, that will make, together with your own lessons, a very useful data-base. We will see now how to reuse the previous lessons, that they be your own or downloaded ones.

All your lessons must be referred to in your configuration file " diaou.conf ", otherwise they will be ignored by software " Diaoulek ". However, the lessons located into the "OWN" directory are automatically taken into account.We will see later the contains of the configuration file.

Then, you have also to create a dictionary of all your entries in alphabetic order. This is automatically done since version 1.3 of the software when some update is necessary, but you can also do it by typing in the command line of the vocabulary window the order " **!dico** ". This order generates the file " diaou_dico ". Here are some lines of such a file :

```
##anken *KK12-3 3995 * Q3 sentiment
##anken #porzh, anken, garv ; porzh an ankenioù garv *K77 7066 * expression
##Ankou skrignet *KE-5 3434 * R2 expression
##ankounac'h *KE-33 2762 * Q2 esprit
##ankounac'haat *KE-36 2567 * verbe société
##annev *K56 5690 * artisan
##annez *K37 7101 * maison
##annezad *K9 5059 * personne société
##annoar *KK3-2 825 * animal ferme
##anoaziñ *K7 3646 * Q2 verbe sentiment
##anoued *KK11-2 2107 * nature température
```

In order to reuse the entry " anken " in one of your lessons, it is enough to make a copy and paste of the line :

```
##anken *KK12-3 3995 * Q3 sentiment
```

into your lesson.

In fact, only the beginning of the line is necessary, you have there the entry you are looking for, here " anken ", followed by the lesson name (alias) where it can be found, here " KK12-3 ". The remaining of the line is not necessary though the list of the tags attached to the data-base entry, here " sentiment " , may be of some use to select a particular word or improve the data-base.

Since version 1.3 of the software, it is possible to select words in the data base in another way. Instead of the order "!dico", you will use the order "!shdic" (**sh**ow **dic**tionary). By simple clicks, you will be able to verify the contents of the data-base corresponding to each entry and you will select the entries that suit you best to make a "Prov" lesson. You will then recuperate in this "Prov" lesson the lines beginning by "##", which are pointers towards the information in your data-base, and you will copy these lines on your lesson. You can also mix both methods.

With what has been said in the present chapter " Composition of a simplified lesson ", you know everything necessary for the redaction of your own lessons. However, it is possible to make more complete lessons and that will be detailed in the following chapter.

A slightly more complete lesson.

The separation line between entries.

For the simplified lessons of the later chapter, the separation line between the Question/Response couples, was reduced to its most simple form. It was composed of a single character : “ # ”. We will see now how we can complete that line and add some more information useful to the management of the data-base or to the display of the entry.

Formation of the entries dictionary.

We have seen that it was very useful to make a dictionary of all the words or expressions of the data base in order to reuse them in other lessons. How is that dictionary made ? In the case of a simplified lesson, the beginning of the “ Question ” is used as reference. Thus for :

```
#
Q>  ur gleizenn ; kleizennoù ; div gleizenn ;
R>  une cicatrice ; pluriel ; deux cicatrices ;
```

the words “ ur gleizenn ” would be retained for the formation of the dictionary. The defect of that method is evident, many dictionary entries will begin by an article ! It is possible to filter the article but it is simpler to impose the word or expression that will be used for the formation of the dictionary. For that purpose, the separation line will be completed by the chosen reference followed by the character “ * ”. For example, in the preceding case we can write :

```
# kleizenn *
Q>  ur gleizenn ; kleizennoù ; div gleizenn ;
R>  une cicatrice ; pluriel ; deux cicatrices ;
```

This couple of “ Question/Response ” will then be referenced under the word “ kleizenn ” in the data-base dictionary. We could have put “ Ref 35 ” or anything else but it is wise to use a reference that is in close relationship with the contains of the article.

Use of commas and semi-colons in referencing the entries.

We can use any character for the referencing of the “ Question/Response ” couple except obviously the “ # ” and “ * ” characters. We can also use commas and semi-colons but those characters have special meanings in versions 1.1 and later of the software. In the the case of multi words, verbs, etc... we can separate these words or verbs by commas as in the following example :

```
#gwennañ, gwennaat *  couleur verbe

<))  aln-k87.ogg  2092031  2253823
Q>   [1] gwennañ ;
     [2] gwennaat ;
R>   [1] blanchir, justifier [blanchir qq en justice] ;
     [2] blanchir, mûrir [pour la moisson] ;
```

This couple of Question-Response will be referenced in the data base dictionary as both “ gwennañ ” and “ gwennaat ”. We will have in the file “ diaou_dico ” :

```
##gwennaat #gwennañ, gwennaat *K87 5959 * couleur verbe
##gwennañ #gwennañ, gwennaat *K87 5959 * couleur verbe
```

The use of the semi-colon will now be explained. When we are referencing a couple of " Question/Response ", the separation line is scanned starting from the first " # " and up to the encounter of a character " * " or " ; "

Let us suppose that we want to reference an expression like " I am hungry like a bear " which is in a lesson whose name is " P7 ". We also want that expression to be referenced under the words " hungry " and " bear " but still let the user know that these words are used in the expression " hungry like a bear ". We will then write :

```
 #hungry, bear ; hungry like a bear *  expression
```

In the file " diaou_dico ", we will have the two references :

```
##bear #hungry, bear ; hungry like a bear *P7 * expression

##hungry #hungry, bear ; hungry like a bear *P7 * expression
```

If we had replaced in the lesson the semi-colon by a comma, we would have had the three following references in the file " diaou_dico " :

```
##bear #hungry, bear, hungry like a bear *P7 1234 * expression

##hungry #hungry, bear, hungry like a bear *P7 1234 * expression

##hungry like a bear #hungry, bear, hungry like a bear *P7 1234 * expression
```

Using the tags.

It was already said that the couples " Question/Response " could receive tags in order to study and review vocabulary with a given tag. These tags are put on the separation line after the character " ***** ". We can have several tags. For example, in the case of " kleizenn " that was given above, we can have :

```
# kleizenn * corps visage médecine
Q>  ur gleizenn ; kleizennoù ; div gleizenn ;
R>  une cicatrice ; pluriel ; deux cicatrices ;
```

Here, three tags " corps " (body), " visage " (face) and " médecine " (medicine) are attached to the entry " kleizenn " (scar). If a tag is used for the first time, it will be added to the tag dictionary.
It is possible to obtain the list, in alphabetic order, of the tags already in use by typing the order " **!ltag** " in the command line of the " vocabulary window ". This command will display the tag list with, for each of them, the number of times it was used in the data-base and the number of words or expressions really studied in both directions QR and RQ. For example for the tag " école " (school), at one time I had the following indication :
* école * nb=33 QR=4 RQ=26
That is to say, in the data-base 33 words or expressions have the tag " école " but only 4 were really studied in the direction Question-Response (Breton to French) and 26 in the direction Response-Question (French to Breton)

The order “ **!ltag** ” creates also the file “ diaou_ltag ” with the same indications. That file can be copied into your lessons if you want to have under your eyes the list of tags already in use. That is convenient when you compose a lesson of your own but this is not mandatory.

The very special tags : “ Qi ” and “ Ri ”.

The tags “ Qi ” and “ Ri ” where “ i ” is an integer are very special. We have already seen, for example into :

```
#
Q>  strafuilhañ (2)
R>  troubler, inquiéter
```

that I find convenient to indicate between parentheses the number of translations expected for the word. In the case of a written interrogation, the indication “ (2) ” will appear because everything written before the marker “ R> ” is displayed on the screen. Obviously, that is not true during an oral interrogation and we have such an interrogation as soon as an audio file is available (and the sound is not off). In order to correct that flaw, it is enough to introduce the tag “ Q2 ” which means that the “ Question ” is waiting for two responses. We will then write something like :

```
# strafuilhañ * Q2 verbe
Q>  strafuilhañ (2)
R>  Troubler, Inquiéter
```

In the above example, we have two tags “ Q2 ” and “ verbe ”. In the case of an oral interrogation, the indication “ (2) ” will appear below the name of the audio file. This means that the question wants two responses (translations).

An example of such a behavior is visible on figure 6 where the Breton word “ fichañ ” is waiting for 3 translations. In the interrogation part (over the line "====") the indication “ (3) ” appears below the name of the audio file “ kk8-2.wav ”. In the verification part (below the line "====") we can find the text in Breton with the indication of the number of expected translations (again !). Then we have the text in French with the three translations.

We must take note that the indication “ (3) ” of the Breton text and the tag “ Q3 ” are independent. It is up to the author of the lesson to make them be the same.

You can also have more than one “ Qi ”. For example the tags “ Q1 ” “ Q2 ” will result in the writing of “ (1) (2) ” in the oral interrogation phase which may be interpreted as : a first words is waiting for a single translation an a second word is waiting for 2 translations.

The tag “ Ri ”, where “ i ” is an integer, was introduced by reason of symmetry when a word in the language of reference has several translations in the language to be learned. In the present state of the software that kind of tag is of no use. However, they would be useful if we had audio files for the language of reference.

Modification of the separation line.

Let us suppose that you want to modify the line of separation between the articles. You can do it without any problem for every thing after the character “*”, that is to say for the tags. If you modify what is between the characters “#” and “*”, you may loose your previous

records for that article. Let us take one example : In lesson “KE-21.txt” we have an article with the expression “ plouz gwiniz” (wheat straw) that was formerly referenced under the word “plouz” only and I want to reference it under both words “plouz” (straw) and “gwiniz” (wheat). For that I need to modify the separation line “**#plouz * ...”** and replace it by the new separation line “**#plouz, gwinizh * ...”**. However I may loose the records for that article because its reference is changed. This will be avoided by retaining the former separation line and make it to be followed by the the new one beginning by the letters “**%#**”. So, we will have :

#plouz * matière plante terre ferme

%#plouz, gwinizh * matière terre ferme

R#paille, froment *

<)) aln-ke-21.ogg 15360 99328

Q> plouz gwinizh (masc.) ;

R> de la paille de froment ;

Remark : The separation line can be followed by several lines beginning by the letters “**%#**”. However this is generally useless because your records are only referenced under the last separation line.

A special line for the inverse dictionary.

In version 1.4 of software “Diaoulek”, we have an inverse dictionary, that is to say a dictionary where the words from the language of reference are translated into the words in the language to be learned. In order to built that dictionary the software selects the lines beginning by “R>” in the articles and do its best to extract the most significant words, while excluding the articles “the, a...” and other short words. The result of this analysis is never perfect. So a special line has been introduced to enforce the reference of the article in the inverse dictionary under given words. In the example above, the line ; “ **R#paille, froment *”** make you sure that the article will be referenced under the French words “paille” (straw) and “froment” (wheat). The syntax of that line is analogous to the syntax of the separation line except for the fact that we have no tags. So it would have been possible to write a line like :

R#paille, froment ; de la paille de froment *

This may help to better define the article but only the two words “paille” and “froment” located before the semi-colon will be used as entries into the inverse dictionary.

The audio files.

It is unlikely that a lay person using software “ Diaoulek ” will create its own audio files. However, it is useful to know that it is possible to point to an audio file in a given lesson by having after the separation line another line such as :

```
<))  ee-2.ogg   2848768   2928128
```

We begin that line by “ **<))** ”, then we have an audio file name, then two numbers indicating the beginning and ending records numbers (frame numbers). As we have generally 40000 frames by second, the beginning and endings numbers are often very large. As a convention of the software, if 0 and 0 are indicated as beginning and endings frames, this means : all the file . So :

```
<))  ee-2.ogg   0   0
```

means all the audio file “ ee-2.ogg ” from beginning to end. Another possibility with the same meaning is to write the line with opening and closing parenthesis characters :

```
<()  ee-2.ogg
```

We can have several audio records, all will be played successively one after the other. Perhaps it will be possible in a future version of “ Diaoulek ”, to make a selection among these files according to the name they have.

Remark : Software “ Diaoulek ” is using the sound library “ Libsndfile ” and so it can read any format recognized by the sound library. For all practical porpoises I am using only “ .wav ” and “ .ogg ” files. The “ .wav ” files are for the originals and the “ .ogg ” files (a free compressed format analogous to mp3) are used for the distribution. Later I may use the “ speex ” format if it is introduced into the audio library. On demand, the “.wav” files can be provided to you.

The configuration files of software “ Diaoulek ”.

Software “ Diaoulek ” has two configuration files : “ diaou.rc ” and “ diaou.conf ”. The first file is in fact a configuration file of the graphic library GTK and the second is specific to the “ Diaoulek ” software.

The configuration file “ diaou.rc ”.

This file is a configuration file of the graphic library GTK, it allows you to adjust the color of the buttons in their various states and the font of the button labels. At least for the first times that you will use this software, it is unlikely that you will need much changes to this file. You can note that the colors are defined by three numbers varying between 0.0 and 1.0. These numbers represent the red, green and blue colors. We can also adjust the fonts by choosing their names and size. Nevertheless the fonts have to exist on your computer and in the selected size or we will fall back on the default font.

The configuration file “ diaou.conf ”.

This file is a configuration file specific to software “ Diaoulek ”. It gives to the software the list of the lessons making its data-base and sets the values of some parameters like the size of the display fonts or the windows default sizes.

The list of the lessons.

The components of the lessons' list must be given between the two flag marks “ **Which_lesson :>** ” and “ **<:** ”. It is better to give only one lesson by line and indicate the absolute or relative path as follow :

```
Which_lesson :>
 ./K-lessons/k86.txt
 ./K-lessons/ks53.txt
 <:
```

However, in order to simplify the writing, when a full set of lessons is available, for example for all the lessons “ ee-i.txt ”, with “ i ” varying between 1 and 10, one can use the

notation “ ee-1->10.txt “. In that case, we have an increase from 1 to 10, but a decrease is also possible by writing : “ ee-10->1.txt ”. It is even possible to have in a file name several variable numbers as below for the “ kk... ” lessons :

```
Which_lesson :>
 ./EE-lessons/ee-1->10.txt
 ./KK-lessons/kk13->1-3->1.txt
 ./K-lessons/ks53.txt
 ./K-lessons/k79->86.txt
 ./KE-lessons/ke-1->76.txt
<:
```

In that example, the notation “ `kk13->1-3->1.txt` ” replaces the full series :

kk13-3.txt, kk13-2.txt, kk13-1.txt, kk12-3.txt, kk12-2.txt, … , kk1-1.txt

Internationalization. Choice of the language.

Version 1.3 of software Diaoulek has introduced internationalization. The language for the communication with the user may be chosen by the order :

```
Lang :>   XXX   <:
```

Where the XXX are two capital letters that may be followed by a number representing a particular flavor for a language. Obviously, the selected language XXX must be present into the software internationalization file “ diaou_internat.txt ”. If for example XX3 is not found, the software will look for XX and if XX is not found, the software will fall back to the default EN (English). At present (February 2014), the internationalization file only contains the EN (English) and FR (French) languages. You may complete this file and send me the completed file for diffusion.

Choice of the fonts.

The fonts in use in the “ lesson ” and “ vocabulary ” windows can be changed by the orders “ Font_lesson ” and “ Font_vocab ”. For example :

```
#
Font_lesson :>   Serif 18   <:
Font_vocab  :>   Helvetica normal 20   <:
#
```

The fonts you want to use must have been set on your computer, otherwise the default fonts will be used instead.

The windows' default sizes.

The default sizes of the various windows can be changed in the configuration file “ diaou.conf ”. You can set the dimension in pixels of the whole frame, vocabulary window + lesson window by the orders “ **Horiz_pix_total** ” and “ **Verti_pix_total** ” as follow :

```
#
 Horiz_pix_total :>   1200  <:
 Verti_pix_total :>   850  <:
```

```
#
```

The width of the vocabulary window can be given in pixels by the order " **Width_pix_vocab** ". For example :

```
# #  Number of pixels for the width of the window displaying
#   the vocabulary (left part of main window) ; example : Width_pix_total/2
#
    Width_pix_vocab :> 580 <:
#
```

Obviously, subtracting " **Width_pix_vocab** " from " **Horiz_pix_total** ", you get the width in pixels of the lesson window.

These sizes are only default values, they are used only at the first start of software " Diaoulek " or after an emergency stop due to a software crash or by a click on the decoration button " × " of the window which contains everything. When using the software, you can adjust the size and position of the windows by dragging the edges or corners. When you leave the software normally by using the " **Quit** " button, the positions and sizes of the various windows are recorded and they will be used at the next start of " Diaoulek ".

Choice of the path to the IDX1 directory.

The software " Diaoulek " creates on its own initiative many files, some of these files are regrouped in directories that are also created if they don't already exist. One of these directories is the " IDX1 " directory where the software will put what it records when you study a lesson. By default, this directory is created in the same directory where the executable " diaou.x " or " diaou.exe " is. However you can change this place by the order :

```
#
    Path_idx1 :> Path_to_dir/IDX1 <:
#
```

This is even necessary when you study several languages because the software must avoid to mix results relative to these different languages. This is checked by " Diaoulek " and will result in an error if two different configuration files have the same " IDX1 " directory.

The path to the " IDX1 " directory is a very important parameter because it determines also the path to other directories such as the " LOG ", " CPY ", " PROV ", " OWN "... directories. This path must be the first thing to be defined in a configuration file.

Study of several languages with software " Diaoulek ".

With software " Diaoulek ", you can study several languages. For that purpose, you only need to have several configuration files. However, one of this configuration file must be called " diaou.conf ", you will chose it for the language that you study most often. You will access and use the other configuration file by the command line, using the order " **!chconf other_conf_file** " where "other_conf_file" is the name of another configuration file which indicates where the lessons for the studied language can be found and where to create the IDX1 directory. As already noted, two configuration files cannot have the same IDX1 directory. The software would complain for the error.

At present time, software “ Diaoulek ” is released with Breton/French lessons and a few English/French lessons which can be reached by the order “ **!chconf diaou_en.conf** ”. These lessons are not for beginners and you will have to make your own lessons on the same model.

You can, obviously, write and study other lessons in other languages thought, at present time, the software is restricted to the use of “Latin 1” characters.

The command line of the “ vocabulary window ”.

Many actions are possible thanks to the command line. A list of all the possible actions can be obtained by writing the order “ **help** ” in that command line. We can for example review the most badly known words, learn new ones, etc... For example the order “ **!worst qr** ” allows you to get the 20 most badly known words and imposes the QR direction. Obviously “ **!worst rq** ” would have set the learning direction to be RQ. Another example : “ **!tag oiseau** ” looks for the 20 most badly known words or expressions with the tag “ **oiseau** ” (bird). The order “ **!ntag oiseau** ” looks for at most 20 words with the tag “ **oiseau** ” that were not studied yet. I let you discover by yourself all the possibilities of the command line by typing the orders “ **help** ” or “ **!help** ” on it. These orders induce the display of a help file in the “lesson window”.At present, we have two help files, one in English and the other in French. The choice is made according to the order “ **Lang** ” in the configuration file. The file in English is the default.

As we have so many possibilities available with the command line, only a few will be detailed in this user's guide. They are the most important ones and by extrapolation, you should not have difficulties to use the other commands.

The orders “ !show ”.

We have a whole group of orders, the “ **!show** ” orders whose purpose is to display and manage the data-base entries. With these orders, we will leave the normal learning mode of the software and we will enter into a management mode where we will be able to select lessons or data-base entries to make other lessons, suppress some of them or return entries to the zero state (unlearned state). A “continue” button will appear at the bottom of the lesson window which, when clicked will allow you return to the normal learning mode of the software.

The order “ !shdic ” to display a dictionary of the data-base entries.

When the number of lessons available in a particular couple of languages becomes more numerous, it is necessary to have tools to list and class all the entries in the data-base. You must be able to rapidly check a particular entry as well as select and reuse it in a new lesson. The order “ **!shdic** ” (**sh**ow **dic**tionary) in the command line will allow you to do that.

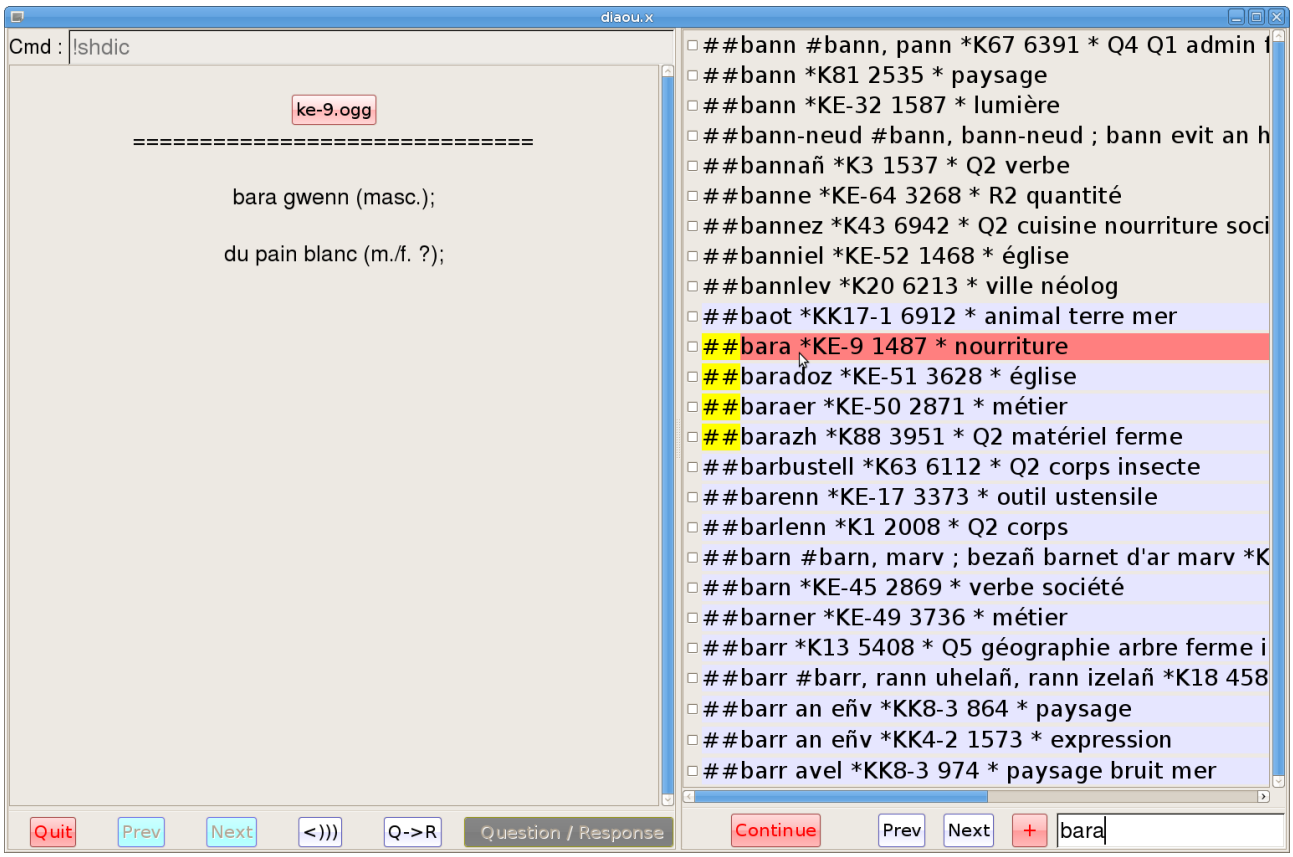
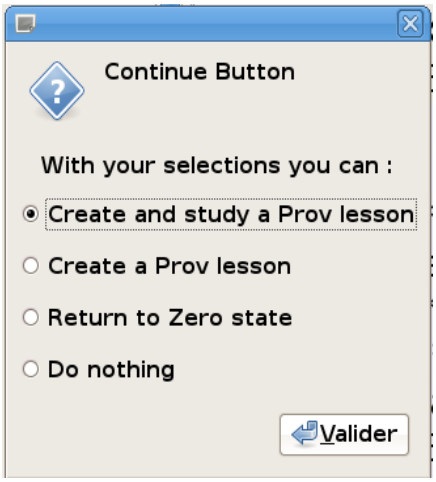


Figure 10 (Diaoulek-14-1.png) *Display of the dictionary of all the data-base entries*.

On the above figure, we can see that we have written “!shdic” in the command line at the top of the vocabulary window. This results in the display of the first page of a list of the data-base entries classed in alphabetic order (at least the gcc alphabetic order !). You can reach the other pages by pressing the buttons “Prev” and “Next” at the bottom of the lesson window. However, in order to look for a particular word, it is often more convenient to write the first letters of that word in the command line at the bottom of the lesson window. In the case of figure 10, we have written “bara” and the software goes directly on the page where the words beginning with “bara” are. The first two characters are highlighted with a yellow background when we have a match. Any line can be clicked and this will results in the display of the article in the vocabulary window. In the case of orders like “!shdic”, the buttons of the vocabulary window, “Prev”, “Next”, “Question/Response” and the command line at the top of this window are inactivated. They are of no use in the present mode. You can select words in the dictionary by a click on the check box of the chosen words. When you have finished your selection, you will leave the “show mode” by pressing the “continue” button at the back of the lesson window. This will open the following dialog box :



This dialog box offers some actions which can be made with the selected words. In particular, you can create with these words a "Prov" lesson or if you consider that you better have to forget them, you can return them to the zero state. The zero state is the state of words that have never been studied

After validation, you will leave the "show" mode and return to the normal "learning lesson" mode. However, you may have wanted to select other words, for example to make another "Prov" lesson or to return some words to the zero state. To be obliged to return to the "show" mode by the order "!shdic" for that purpose is annoying. You can avoid that by a click on the "+" button at the back of the lesson window (see figure 10). The "+" button will display the same dialog window as the "continue" button but you will not leave the "show" mode.

The order " !shtag " to display a dictionary of tagged data-base entries.

In the preceding paragraph, we have generated a complete dictionary of the data-base entries. It is also useful to restrict the dictionary to the entries with a given tag name. This can be done by the order "!shtag" (show tag) which must be followed by a tag name for example "!shtag oiseau" will display a dictionary of the data-base entries with the tag "oiseau" (bird). You can, as for the order "!shdic", click on words and select some of them for a "Prov" lesson or a return to the zero state.

The order " !shcid " to display an inverse dictionary of the data-base entries.

The dictionary displayed by the order "!shdic" is a list of words in the language to be learned. In version 1.4 of the software, we have an inverse dictionary. It is an ordered list of words in the language of reference. These words are the translations of the words studied in your lessons. In the most recent lessons, each article has a special line starting with "R#" which is used for the creation of the inverse dictionary. That was not the case of the first lessons, for them, the software will do its best to extract significant words from the "R>" lines (the Response part of the article).

You can display the inverse dictionary by the order "**!shcid**". You will remark that "cid" is "dic" written in reverse order. With the check boxes, you can select articles of your data-base and study these articles in a "Prov" lesson.

The order " !shgat " to display an inverse dictionary of tagged data-base entries.

It can also be useful to limit the inverse dictionary to a given tag. This is possible by the order "**!shgat**" followed by a tag name. You will remark that "gat" is the word "tag" written in reverse order.

The order " !shless " to display the entries in a given lesson.

We can also restrict our entries to those in a given lesson. This is done by the order " !shless " which must be followed by a number or by the alias of the lesson. The number is the rank of the lesson in the list of recognized lessons. The rank of the lesson may vary, it depends from your configuration file, the alias is a unique characteristics of the lesson. For example " !shless EE-1 " will list the contains of the first complete lesson and the order " !shless 1 " will do the same thing if that particular lesson is the first one in the list of lessons in your configuration file "diaou.conf".

The order " !shprov " to display the list of all provisional lessons.

We have already encountered this order when we spoke of the provisional lessons. This order display the list of the provisional lessons ranked by order of decreasing length of life span. By a simple click on the lesson name you have a list of the words in the lesson. You can also select these lessons to study, regroup or even delete them.

The order " !ccdb " to check and correct the data-base.

Your data-base is a dynamic element, which has to maintain the list of all the words and expressions that are part of it. We have also to known in what lesson and where in the lesson we can find the words and expressions. In the data-base, we must also have the recording of all your learning results. The data-base must adapt itself when you modify, introduce or remove lessons. As a consequence, the data-base is always varying and, as in Nature with the genetic code, any error in the data-base will propagate and be amplified. The software " Diaoulek " contains elaborate mechanisms to maintain, verify and correct the data-base. These mechanisms are automatic and transparent for the user. However, in some cases, it can be useful to force a verification/correction to be made. That will be the case, for example if you observe some mixing between the " Questions " and the " Responses " when you study a lesson. Then, to correct the problem you will type, in the command line of the vocabulary window, the order " **!ccdb** " (which means " Check and Correct Data-Base "). That order forces a correction of the complete data-base. Normally, you should not lose any element of the previous records. Only some rare entries that are truly irrecoverable should be reseted to zero.

The order " **!ccdb** " is fast and efficient. My advice is to use it systematically in some somewhat peculiar cases :

1) If you share the same data-base between two not very compatible operating systems like MS-Windows and Linux. In that case the dates of modification of the files are not always well transmitted and that may induce problems when you modify the lessons.
2) If you transmit lesson files with an USB key. In that case the transmission of the dates is often erroneous. This induces errors when the lessons are not new ones. You have to force an updating of the data-base.
3) If you are using the order " !synchro " in order to synchronize the data-base between two computers. That can be useful, for example after a vacation if you have continued to use " Diaoulek " on a portable...

The order " !import " to import lessons, sound files and the data-base.

The order " !import " has been introduced in version 1.2 to ease the software upgrading by

making automatic the copy of the lessons and data-base of a previous version. When you upgrade your software, the quickest way is to copy the executable into the same directory where your previous executable was and use it in lieu of the older one. Then a “ !ccdb ” order will check and perhaps adapt your data-base to your new “ Diaoulek ” version. This method will work for Windows when the “ Diaoulek ” versions are compatible (this is the case for 1.2 and 1.3 versions). In all other cases you must load the new version of the software in a new directory and import your lessons and their sound files as well as the data-base from the former “ Diaoulek ” version's directory. For that purpose, you will use the order “ !import ” which must be followed by the absolute path towards your older configuration file. For example :

```
!import /home/user/D1-0/diaou.conf
```

or on a Windows OS :

```
!import C:\Program Files\D1-0\diaou.conf
```

The “ Diaoulek ” configuration file of the former version will be scanned, the names of the lessons will be extracted and these lessons and their associated sound files will be copied into your new “ Diaoulek ” directory. Your former data-base will also be copied. However, if a lesson already exists into your new “ Diaoulek ” directory, it will not be imported. That is why the order “ !import ” should be used before any other action as soon as you have obtained your new executable.

The order “ !import ”, once it has imported all your lessons and the data-base, closes your “ Diaoulek ” session. You will then start another “ Diaoulek ” session and execute a “ !ccdb ” order. This will check, correct and adapt the copy of your data-base. You may then continue to use software “ Diaoulek ” as usual.

Remark 1 : The order “ !import ” is also useful if you want to make a copy of your lessons and data-base into another directory for example on a USB key.

Remark 2 : The order “ !import ” makes a copy of your recorded results in a IDX1 directory whose contains will change when you continue to learn your lessons. It may however be interesting to have a snap-shot of your recorded results at the time of the duplication. This may allow you for example to add your activity recorded on a USB key to your main computer when you are coming back from vacation. For that purpose, the order “ !import ” makes in fact two copies of the recorded results, one in a IDX1 directory (whose contains will change) and the other into a IDX0 directory (whose contains will not change).

The orders “ !synchro ” and “ !synchro0 ”

These two orders must be followed by the absolute or relative path towards some IDX1 directory.

Let us suppose that you are in the following situation, you study a language on your main desktop computer at home. Then, you have to leave for vacation time. You make a copy of software Diaoulek on a USB key that will go with you. In your USB key, you create a copy of the lessons and of your IDX1 directory that are on your main computer by the order “ !import ”. You are now able to use this USB key on another computer with a compatible OS (Windows on Windows or Linux XXX on Linux XXX). Coming back from vacations, you continue your studies on your desktop computer for a few days. Then you remember that it will be nice to take into account your results recorded on the USB key during your vacation. For that, on your home computer, you will write on the command line the order

“**!synchro0** path_to_IDX1_dir_on_USB_key”. This will add the difference IDX1-IDX0 of your USB key directories to your present IDX1 directory. By that order, you have imported exactly your work during your vacation even in the case where you have had also some activity on your home computer.

Let us consider now a more complex situation. After your return from vacation and the update of your data-base by the order “!synchro0”, you have to make an unexpected voyage for professional reasons. You have no time to create another USB key with software Diaoulek. Then you take with you the USB key created for your vacations and you continue to study with it. When you are back home, you must not upgrade your data-base on your home computer by the order “!synchro0” because in so doing, you would take into account two times your vacation work. Here, instead of “!synchro0”, you will use the order “! synchro” which does not take into account a zero state. The order “!synchro” is not as powerful as the order “!synchro0” however, it will only change the records if the numbers on the USB key are greater than those on the home computer. Thus, some of your work on the USB key may be overlooked.

Known “ bugs ” in software “ Diaoulek ”.

Any software released in open nature goes with its swarm of “ bugs ”. The present project makes no exception to the rule. Among the known “ bugs ”, we can quote :

1) Under Linux/Ubuntu, with and even without Alsa, a long enough record makes the screen to darken. This does not seem to be the case with Mint 13.
2) The main button can freeze. That may not be due to “ Diaoulek ”, however. In any case, it is enough to click two times on one of the two buttons at the left of the main one and the software will continue its job. The problem does not seem to happen on Linux Mint 13.
3) The installation under Linux must be done by trial and error. That is the common problem for softwares on Linux when there are no ready made packages for the specific distribution.
4) There are many mistakes in the lessons and the audio files are not perfect. That can be improved with the participation of everybody.

Some advices on how to use software “ Diaoulek ” 1.4.

As a conclusion, here are some advices on how to use version 1.4 of software “ Diaoulek ”. As it is true for any software, it is only a tool and you have to learn how to use it best. Because of the numerous possibility of software “ Diaoulek ”, everybody can use it in a way that will be best suited to his needs and his remembering possibilities. Some general advices may however be useful. The key to success when learning a language is everyday repetition. A little every day allows you to learn, much from time to time is of no value. In order to learn a language, you have first to listen to it. The ear have a memory of its own, young children are able to understand before they speak. When we are a grown up person, it is possible for us to speak as we have studied in books with grammar and we can chose our words. To do mistakes when speaking is not too unforgivable (except at school !), but you still have to understand the reply. For that, your ear would have to be used to various kinds of voices, masculine, feminine, young children voices, and also to various accents and

speech rates. Perhaps, one day, software “ Diaoulek ” will offer you that possibility if various persons are releasing audio files. For the moment, you will have to do with only one voice in Breton, it is in any case better than nothing...

When learning a language, my advice is to use a good method, a not too difficult and very progressive one. Many are available, but anyway, you will have to use the one selected by your professor or by yourself if your are an isolated student. Then, you will use software “ Diaoulek ” to learn and review the vocabulary of your method's lessons. You will have to write your own personal lessons for software “ Diaoulek ”. You can do that in two different ways. The first way is to do everything manually like in former versions of the software. Lessons, as explained earlier, are plain text files that may be very simple to write. In the Breton → French case, you will use the data-base made from all the published lessons and the writing of your own lessons will, most often, be reduced to simple copy and paste actions, at least if you are a beginner. As the simplest and most frequently encountered vocabulary is already in the data-base, since version 1.3 you have a second and more automatic possibility to generate a lesson. You will use the orders “ !shdic ” and “!shcid” to display a dictionary of all the data-base entries. By a simple click of your mouse on a check box you can select entries and with the selected entries you can generate a lesson with a limited life span (the “Prov” lessons). However, you have limitations with the use of provisional lessons. You cannot modify these lessons and introduce your own words, you cannot change the file names and their alias. That is why in version 1.4 of the software, the “Own” lessons which don't have these limitations have been introduced. You may consider the “Own” lessons which are located into an “OWN” directory, as an help to create your own personal lesson. At each start, the software is visiting the “OWN” directory and so it is not necessary to declare your “Own” lessons into the configuration file.

In order to learn your personal lessons, you will call them in the command line by their ordering numbers or by their aliases or still by a selection from the order “!shown” which is very much like the order “!shprov” which is for the management of the provisional lessons. Don't forget to learn your lessons in both directions QR and RQ.

The review of the vocabulary that you have already well studied will be done by the use of the orders “ !worst qr ” and “ !worst rq ”. You can have as many “ !worst ” lessons that you may want, one following the other. Of course, you will be going successively towards better known words, at least in theory. You can stop the learning of a lesson after a first pass over each word (first run), or you can continue for a better learning. You have many kinds of compound lessons computed according to different criteria. They will allow you to criss-cross through your previously learned vocabulary. The main drawback of the methods used at school is that we keep learning new words and let the review of that same, so difficultly (and provisionally) learned, vocabulary to be done by chance encounters in texts. Thanks to its records, software “ Diaoulek ” allows you to review your vocabulary in an optimal fashion. Forgetting is not bad, I even believe that this is an integral part in the learning process. Too bad, however that it is the easiest part of the job, but it is not difficult to learn again what was already well studied. If you use it regularly, software “ Diaoulek ” will help you to do that !

The best method to learn new vocabulary is probably to review words often at short time intervals during the learning phase and then review these words from time to time with longer time intervals to refresh the memory. The software “ Diaoulek ” has a mechanism to help you do just that. Each time you have learned a lesson, you can generate with the badly known words a “ Prov ” lesson. You can select some “ Prov ” lessons, sum and study them

and again generate another “ Prov ” lesson with the still badly known words. With this process, a word will remain in the “ Prov ” lessons as long as it is not well known. The memory refreshment can, after that, be done by the “ !worst ” lessons. It is up to the user to find the right combination of “ Prov ”, “ !worst ” and personal lessons best suited to his needs and to his memory and to alternate between both learning directions Q->R and R->Q.