3ET – An Automatic Tool for Grammar Training

Claus Tøndering
Ezer IT Consulting
claus@ezer.dk

Reference: Claus Tøndering, "3ET – An Automatic Tool for Grammar Training", *Hiphil* 6 [http://hiphil.see-j.net] (2009). Accessed DD.MM.YYYY.

ABSTRACT

3ET is a computer program that automatically generates grammar questions for a foreign language. Students of a language can use 3ET to generate questions to train their knowledge of the morphology of the language, and teachers can use 3ET to create quizzes to test the students' knowledge. 3ET uses the Emdros text database system and is primarily intended for the study of biblical Hebrew and Greek.

1. BACKGROUND

A number of corpora have been analysed in great detail by experts of various languages. In some cases, detailed grammatical information about every single word in the corpus exists, giving information about the word's number, gender, case, tense, mood etc. This type of detailed analysis is available for both the Hebrew Old Testament and the Greek New Testament.

These thoroughly analysed and annotated corpora exist in electronic form. Some are available free of charge, others are sold by various organisations.

The idea behind the 3ET program is to use the grammatical information is such a corpus to generate grammar exercises for students of the particular language.

More information about the project can be found on http://3bm.dk/index.php?p=82.

2. THE EMDROS TEXT DATABASE

A particularly useful database system for storing and retrieving annotated text is the *Emdros* database engine[1] developed by Assistant Professor Ulrik Sandborg-Petersen of Aalborg University. A short introduction to Emdros for linguists is found in (Petersen, 2004).

For detailed information about Emdros, the reader is referred to the Emdros web page mentioned in footnote 1. Here, only a very brief description of the system will be given.

[1] Available at http://emdros.org.

Emdros divides a text into *objects*. Typical objects are *word, clause,* and *sentence.* For biblical texts, objects such as *book, chapter,* and *verse* are also used. Each occurrence of the smallest object, typically a word, is identified by a number, called a *monad* in Emdros terminology.

As an example, consider this text:

"The boy, who had red hair, was sitting on the floor."

This sentence consists of 11 words. We can also identify two clauses, "The boy ... was sitting on the floor" and "who had red hair".

Emdros assigns a monad (a positive integer) to each word object:

Monad	Word	Monad	Word
1	The	7	was
2	boy,	8	sitting
3	who	9	on
4	had	10	the
5	red	11	floor.
6	hair,		

Additionally, sets of monads are used to identify *clause* objects: Monads { 1-2, 7-11 } identify one clause, and monads { 3-6 } identify another. Finally, the monads { 1-11 } identify a *sentence* object.

Associated with each object are a number of *features* that describe various characteristics of the object. For a word object, typical features could be *part of speech, gender, number, tense,* and *mood.* In the above sentence, word object 4 (the word "had") could, for example, have these features:

Feature	Value
Text	"had"
Part of speech	Verb
Tense	Past
Number	Singular
Person	3rd
Lexeme	"have"

Similarly, the two clause objects could have a feature called *type* with the values *main* and *subordinate*, respectively.

The exact set of objects and features available in a database is entirely up to the person who creates the database.

Emdros comes with a query language, called MQL[2], that allows a user to search a corpus for objects with various features. MQL queries can be quite simple, such as “find all verbs in the past tense”, or very complex, such as “find all sentences containing a singular pronoun, followed by at most three words, followed by a verb in the present tense, except cases where the verb is derived from ‘to be’”.

The exact syntax of MQL queries can be quite arcane and is beyond the scope of this paper. The two examples in the previous paragraph might look like this:

```
[word part_of_speech=verb AND tense=past]
```

and this:

```
[sentence [word part_of_speech=pronoun AND number=singular]
    .. <= 3 [word part_of_speech=verb AND tense=present
    AND lexeme<>"be"]]
```

3. THE 3ET PROGRAM

3ET is short for “Ezer’s Emdros-based Exercise Tool”[3]. The purpose of the program is to use the contents of an Emdros database to automatically generate grammatical questions for students of a language. 3ET may, for example, take the sentence “The boy, who had red hair, was sitting on the floor” and ask a student to identify the tense of the verb “had”.

3ET can be used by students of a foreign language who want to drill themselves in the knowledge of the language, or it may be used by teachers who want to create tests for their students.

An exercise is generated by going through a number of steps:

1. *Specify the Emdros database that contains a relevant corpus.* Typically, a corpus will be the Hebrew Old Testament or the Greek New Testament.
2. *Specify the “universe” for the exercise.* The “universe” is a subset of the entire corpus. For example, a teacher or student may specify that an exercise should only use sentences from the Pentateuch.
3. *Specify the characteristics of the sentences used in the exercise.* This amounts to a search query in the Emdros database. Such a query might, for example, be: “Find sentences containing a verb that is not of the qal stem.”
4. *Specify the “quiz objects” within the sentences.* This is another Emdros search query which looks for the words that are to be the subject of the exercise. This could, for example, be: “Find verbs.”

[2] Mini Query Language. (Quite a misnomer, since this is very powerful language.)
[3] Ezer is the name of the author’s company.

5. *Specify the object features that should be shown to the student and the features that the student should provide.* For example, the program may give the student the stem of a particular verb and ask the student to identify the tense.

Based on these specifications, 3ET will generate an exercise.

3.1. EXAMPLE: BUILDING AN EXERCISE

The following screen shots show how the above steps are carried out in the program.

When 3ET is started, it shows a window with the 3ET logo.

Step 1: Specify the Emdros database that contains a relevant corpus.

From the File menu we choose the “New template” item.

Figure 1 shows that in this particular installation, 3ET has access to two databases, a Hebrew Old Testament from the WIVU[4], and a Greek New Testament based on the Tischendorf text.

For this example, we choose the Old Testament database.

[4] *Werkgroep Informatica* of the *Vrije Universiteit* in Amsterdam. The integration of this database has been done with the help of Associate Professor Nicolai Winther-Nielsen of the Copenhagen Lutheran School of Theology, to whom the WIVU has kindly granted permission to use this database for teaching purposes. See (Winther-Nielsen, 2009).

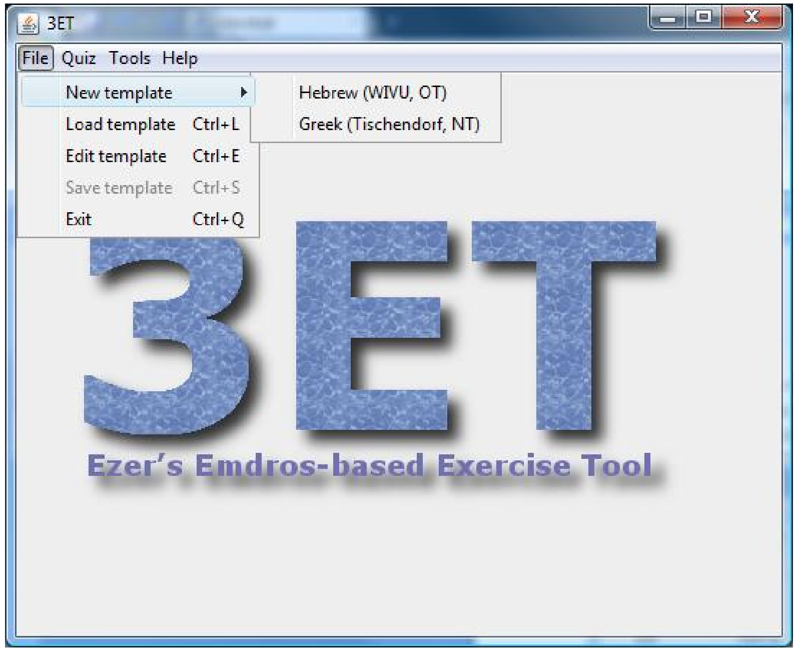
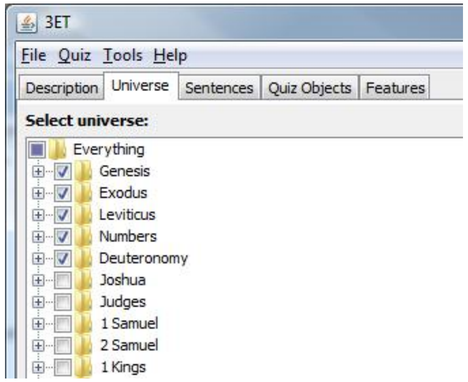


Figure 1. Select Emdros database

When the program has opened the database, it presents a number of tabs at the top of the window. Under these tabs we specify the characteristics of the exercise.

Step 2: Specify the "universe" for the exercise.

Figure 2 shows the contents of the "Universe" tab. Here we can choose the books, chapters, and even the verses, from which out exercises should be built. In this example we choose the Pentateuch.



Select universe:

- Everything
  - [✓] Genesis
  - [✓] Exodus
  - [✓] Leviticus
  - [✓] Numbers
  - [✓] Deuteronomy
  - [ ] Joshua
  - [ ] Judges
  - [ ] 1 Samuel
  - [ ] 2 Samuel
  - [ ] 1 Kings

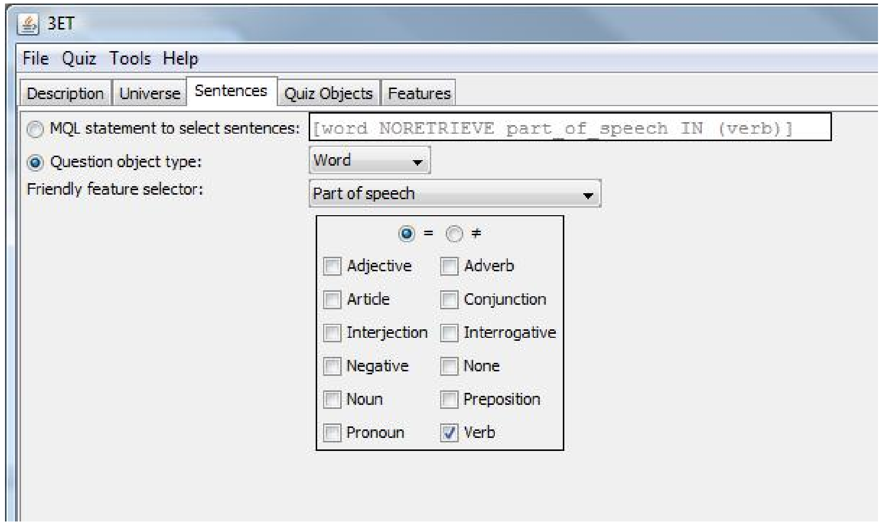
Figure 2. Selecting a universe.

Step 3: Specify the characteristics of the sentences used in the exercise.

Figure 3 shows the contents of the "Sentences" tab. Here we specify the characteristics of the sentences we will use for our exercises. Basically, such a specification takes the form of an MQL statement that selects sentences. 3ET does indeed allow the user to specify the relevant MQL statement directly by selecting the option "MQL statement to select sentences". However, most people will probably find the MQL syntax too unwieldy, and therefore 3ET provides a more user-friendly interface which can build simple MQL statements automatically.

Figure 3 and Figure 4 illustrate this interface. In our example, we are interested in sentences that contain *words* that are *verbs* whose stem is not *qal*. First, we select the object type "Word". Next, we select a feature of the word object; here "Part of speech" is selected. Finally, permissible values of this feature are indicated by selecting the = sign and ticking the value "Verb". As we go through these steps, the system automatically builds an MQL statement. By now, this statement reads

```
[word NORETRIEVE part_of_speech in (verb)]
```



MQL statement to select sentences: [word NORETRIEVE part_of_speech IN (verb)]

[✓ Question object type:] Word

Friendly feature selector: Part of speech

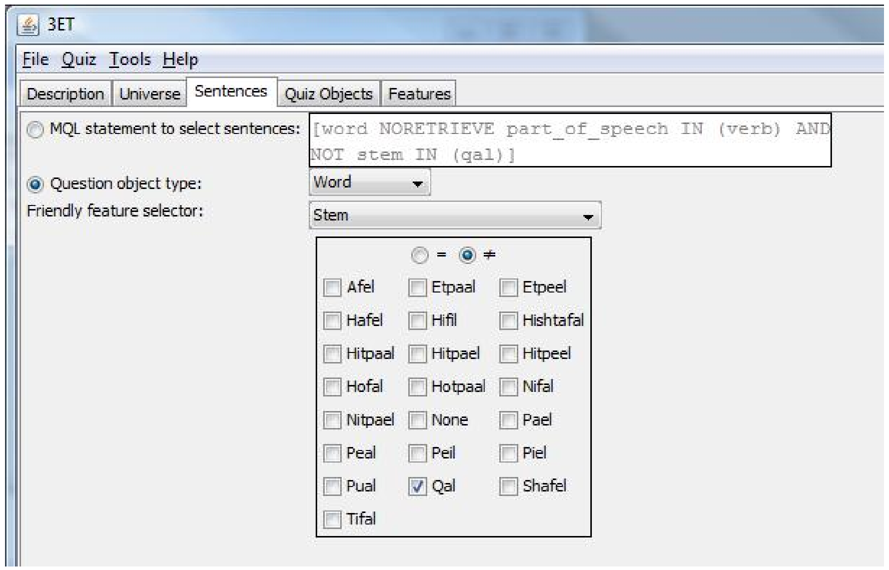
[✓ =] ≠

[ ] Adjective	[ ] Adverb
[ ] Article	[ ] Conjunction
[ ] Interjection	[ ] Interrogative
[ ] Negative	[ ] None
[ ] Noun	[ ] Preposition
[ ] Pronoun	[✓] Verb

Figure 3. Specifying the exercise sentences

So far we have specified sentences containing words that are verbs.

We proceed in Figure 4 to select “Stem”, select the ≠ sign, and tick the “Qal” box. In this manner we indicate that the stem of the word must not be qal. If we wanted to exclude additional stems, we could do so by ticking off each one.



MQL statement to select sentences: [word NORETRIEVE part_of_speech IN (verb) AND NOT stem IN (qal)]

[✓ Question object type:] Word

Friendly feature selector: Stem

= [✓ ≠]

☐ Afel	☐ Etpaal	☐ Etpeel
☐ Hafel	☐ Hifil	☐ Hishtafal
☐ Hitpaal	☐ Hitpael	☐ Hitpeel
☐ Hofal	☐ Hotpaal	☐ Nifal
☐ Nitpael	☐ None	☐ Pael
☐ Peal	☐ Peil	☐ Piel
☐ Pual	✓ Qal	☐ Shafel
☐ Tifal		

Figure 4. Specifying the exercise sentences (cont.)

The system continues to build the MQL statement, which now reads:

```
[word NORETRIEVE part_of_speech in (verb) AND
    NOT stem IN (qal)]
```

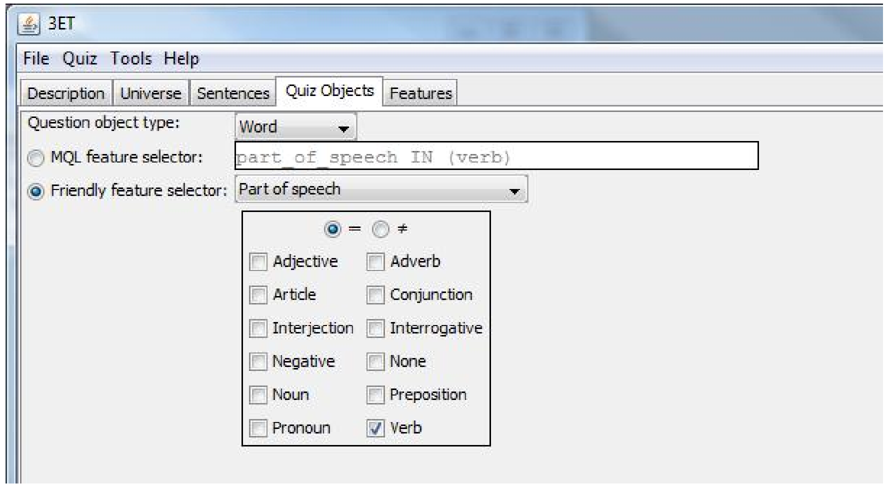
3ET will further embed this in a `[sentence ...]` selector, yielding a final MQL statement which reads:

```
[sentence [word NORETRIEVE part_of_speech IN (verb) AND
    NOT stem IN (qal)]]
```

This statement says that we are looking for sentences that contain words whose *part_of_speech* features is *verb* and whose *stem* feature is not *qal*. (The word “NORETREIVE” speeds up the search. It has no effect on the outcome.)

Step 4: Specify the “quiz objects” within the sentences.

This is very similar to the previous selection. Figure 5 shows that once again we choose a word object and indicate that the word must be a verb. The order of the options is slightly different here. This is because our possibilities are more limited. In the previous step, we might have wanted to specify a very complex MQL statement that involved several words or other objects. In this step, we deal with only one type of object, in this case a *word*.



Question object type: Word

MQL feature selector: part_of_speech IN (verb)

[✓ Friendly feature selector:] Part of speech

[✓ =] ≠

☐ Adjective	☐ Adverb
☐ Article	☐ Conjunction
☐ Interjection	☐ Interrogative
☐ Negative	☐ None
☐ Noun	☐ Preposition
☐ Pronoun	✓ Verb

Figure 5. Specifying quiz objects

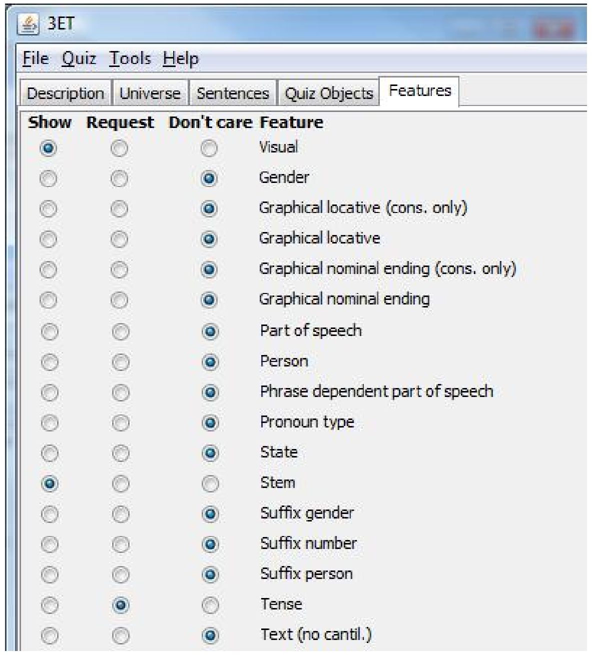
We have now specified that we want to ask the student about all words that are verbs. We therefore move to the final step:

Step 5: Specify the object features that should be shown to the student and the features that the student should provide.

Figure 6 shows the contents of the “Features” tab. Here we have a list of all the features available for word objects. For each feature we specify if we want to show this feature to the student or if the student should identify the value of the feature.

In this example, we specify that the student should see the “Visual” and “Stem” features and identify the “Tense” of the word. The “Visual” feature is the actual text of the word.

The exercise is now ready.



Show	Request	Don't care	Feature
●	○	○	Visual
○	○	●	Gender
○	○	●	Graphical locative (cons. only)
○	○	●	Graphical locative
○	○	●	Graphical nominal ending (cons. only)
○	○	●	Graphical nominal ending
○	○	●	Part of speech
○	○	●	Person
○	○	●	Phrase dependent part of speech
○	○	●	Pronoun type
○	○	●	State
●	○	○	Stem
○	○	●	Suffix gender
○	○	●	Suffix number
○	○	●	Suffix person
○	●	○	Tense
○	○	●	Text (no cantil.)

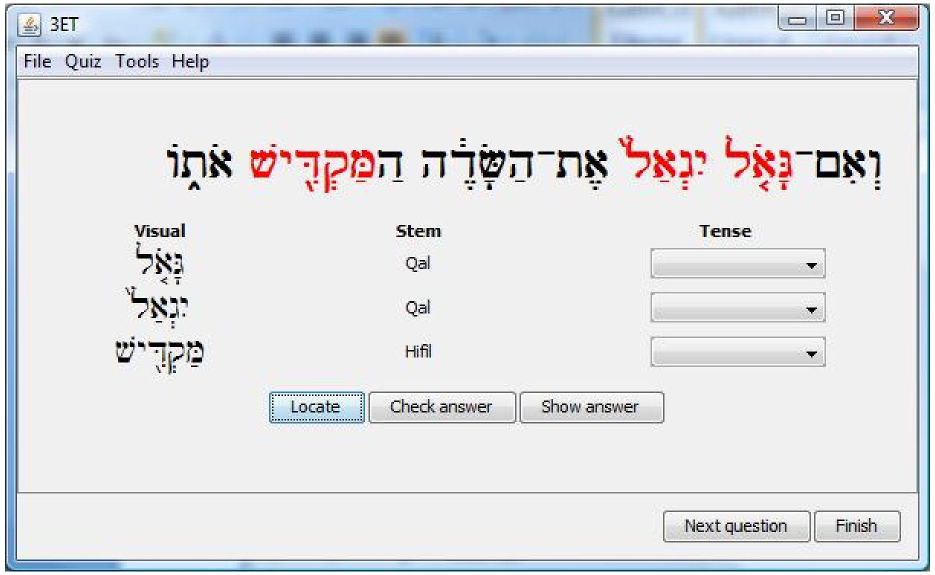
Figure 6. Specifying quiz features

These steps may be performed by a teacher preparing a test for students or by a student creating an exercise for personal use. At this point the exercise may be saved for later use, or it may be executed immediately.

3.2. EXAMPLE: RUNNING AN EXERCISE

From the “Quiz” menu a student chooses to run the exercise. 3ET now selects a random sentence that meets the specified criteria: The sentence must come from the Pentateuch and it must contain a verb that is not of the qal stem.

The first random sentence chosen by 3ET may be the one shown in Figure 7.



וְאִם־גָּאֹל יִגְאַל אֶת־הַשָּׂדֶה הַמַּקְדִּישׁ אֹתוֹ

Visual	Stem	Tense
גָּאֹל	Qal	
יִגְאַל	Qal	
מַקְדִּישׁ	Hifil	

Figure 7. A sample question

A curious student may wonder where this sentence comes from. Pressing the “Locate” button will reveal that the sentence is taken from Leviticus 27:19.

We specified that the quiz objects should be verbs. In Figure 7 the three verbs in the sentence are highlighted in red. Below the sentence the three quiz features (“Visual”, “Stem”, and “Tense”) are found. The first two features are presented to the student; but the student is required to identify the tense of each word. The “Tense” column contains drop-down lists (Figure 8) giving all the possible values of this feature.

When we specified sentences in Figure 4 we indicated that the sentences must contain verbs that are not of the qal stem, and as we can see here, the sentence contains a verb of the hifil stem. But when we specified the quiz objects in Figure 5, we indicated that all verbs should be shown; therefore the question also shows two qal verbs.

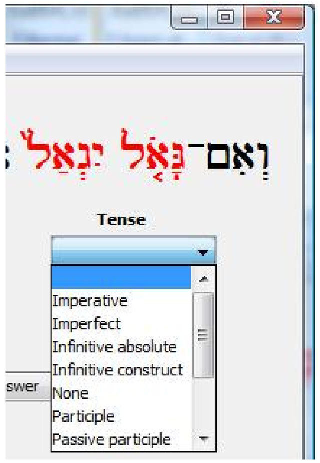
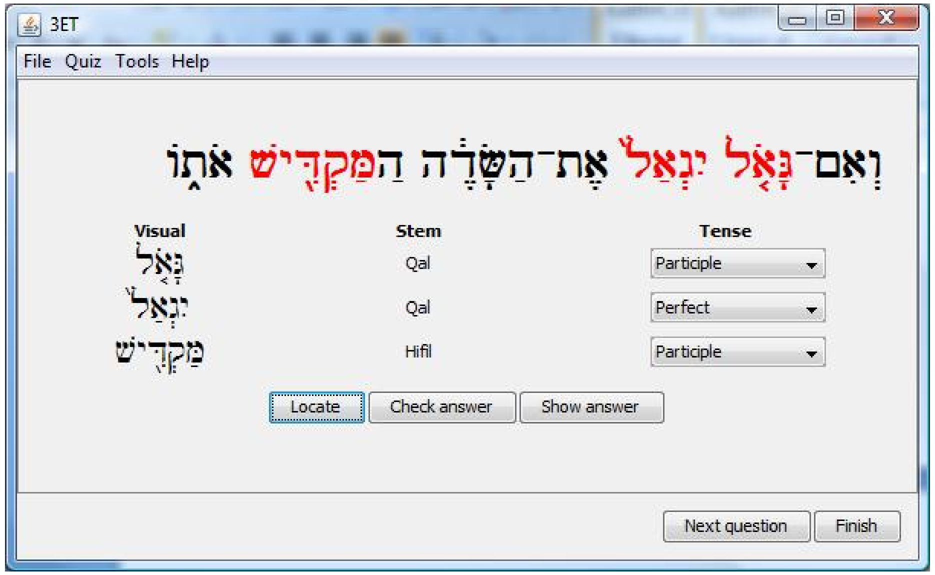


Figure 8. Selecting a tense

Let us assume that the student chooses ”Participle”, ”Perfect”, and ”Participle” as the respective tenses for each verb[5]. See Figure 9.



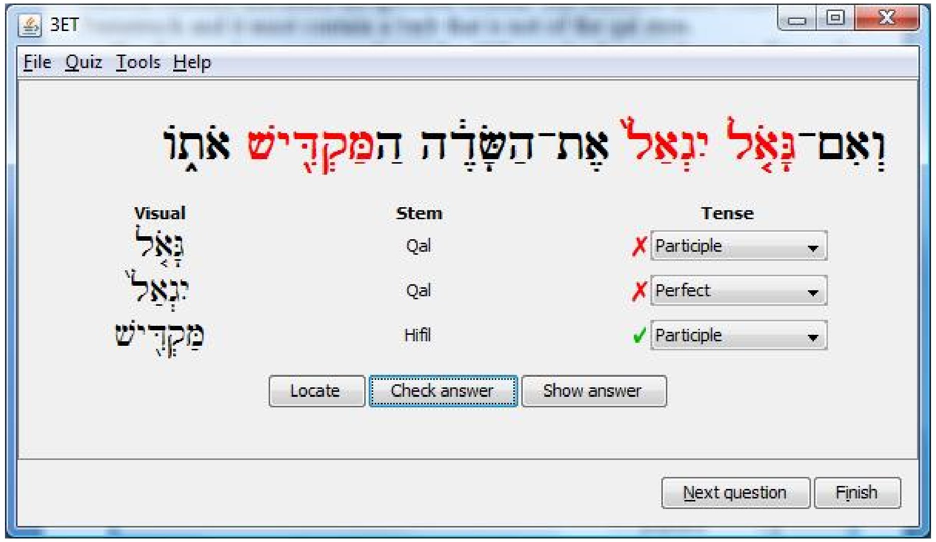
וְאִם־גָּאֹל יִגְאַל אֶת־הַשָּׂדֶה הַמַּקְדִּישׁ אֹתוֹ

Visual	Stem	Tense
גָּאֹל	Qal	Participle
יִגְאַל	Qal	Perfect
מַקְדִּישׁ	Hifil	Participle

Figure 9. The questions have been answered (wrongly)

[5] It may be argued that a participle is not a tense, but that is how the WIVU database handles the verb forms.

When the student clicks on the “Check answer” button, 3ET checks the tenses against the contents of the database. The result is shown in Figure 10.



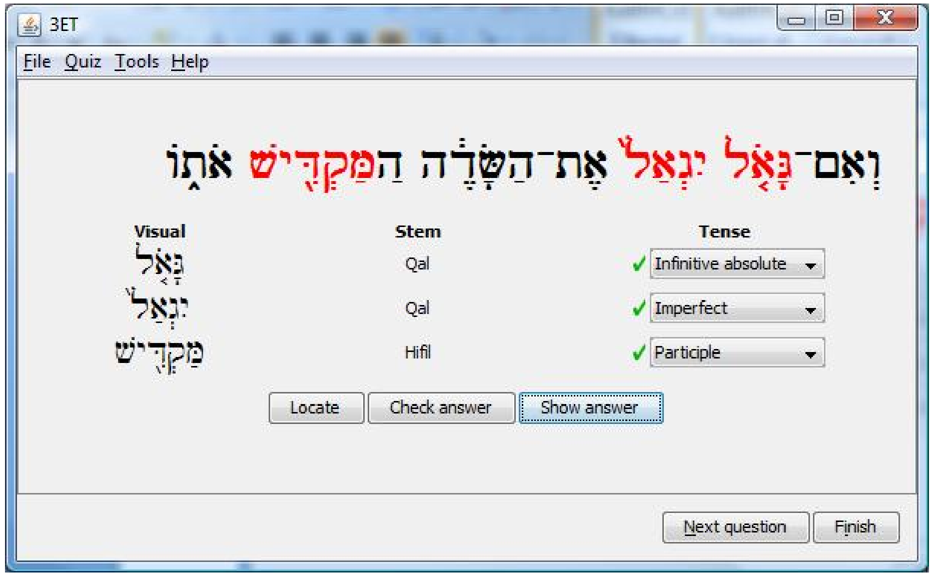
וְאִם־גָּאֹל יִגְאַל אֶת־הַשָּׂדֶה הַמַּקְדִּישׁ אֹתוֹ

Visual	Stem	Tense
גָּאֹל	Qal	✗ Participle
יִגְאַל	Qal	✗ Perfect
מַקְדִּישׁ	Hifil	✓ Participle

Figure 10. Checking the answer

The two first answers were wrong, the third one was correct.

The student may now modify the answers or simply give up and press the “Show answer” button, in which case the system provides the correct answer (Figure 11).



וְאִם־גָּאֹל יִגְאַל אֶת־הַשָּׂדֶה הַמַּקְדִּישׁ אֹתוֹ

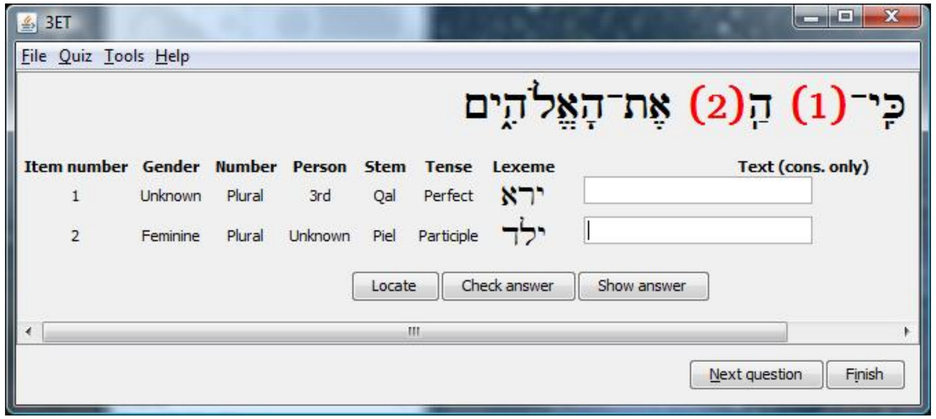
Visual	Stem	Tense
גָּאֹל	Qal	✓ Infinitive absolute
יִגְאַל	Qal	✓ Imperfect
מַקְדִּישׁ	Hifil	✓ Participle

Figure 11. The correct answer

After this, the student may press the “Next question” button to get another random sentence from the universe.

3.3. WRITING TEXT

Occasionally, an exercise may require the student to specify actual text. In this case, the behaviour of 3ET changes in two ways as illustrated in Figure 12.



כִּֽי־(1) הַ(2) אֶת־הָאֱלֹהִים

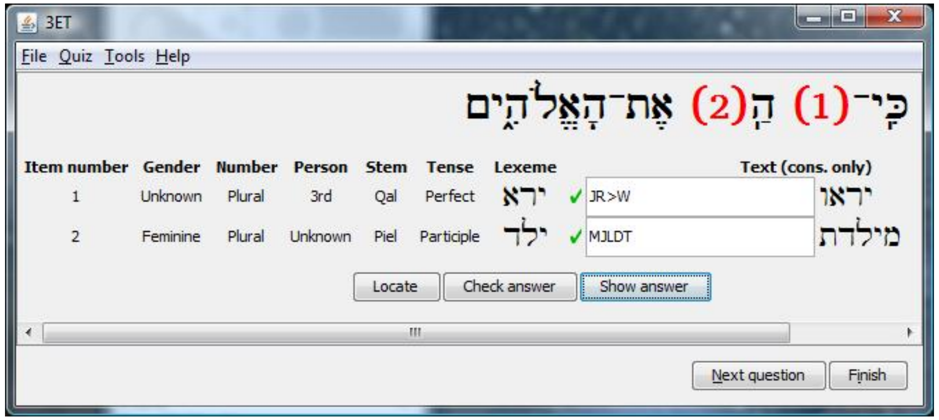
Item number	Gender	Number	Person	Stem	Tense	Lexeme	Text (cons. only)
1	Unknown	Plural	3rd	Qal	Perfect	ירא	
2	Feminine	Plural	Unknown	Piel	Participle	ילד	

Figure 12. Writing text (Exodus 1:21)

In this example the exercise provides the gender, number, person, stem, tense, and lexeme of a verb and asks the student to provide the correct form of the verb. For simplicity, the student is only asked to provide the consonant text.

Since the actual form of the verb is part of the sentence, 3ET must hide this from the student. Therefore, the first change we note is that the relevant words are replaced by numbers in the text and the numbers are repeated in the leftmost column in the question table.

Secondly, instead of a drop-down list with multiple choices the student has a text-entry field for the answer. Students with a Hebrew keyboard can enter the Hebrew characters directly, but for most people this is not the case. Instead a simple character-by-character substitution allows the student to type the Hebrew text using Latin characters; see Figure 13. As each Latin character is typed, the corresponding Hebrew character appears to the right of the text field. A final click on the “Check answer” verifies the answer.



כִּי־(1) הָ(2) אֶת־הָאֱלֹהִים

Item number	Gender	Number	Person	Stem	Tense	Lexeme		Text (cons. only)	
1	Unknown	Plural	3rd	Qal	Perfect	ירא	✓	JR>W	יראו
2	Feminine	Plural	Unknown	Piel	Participle	ילד	✓	MJLDT	מילדת

Figure 13. Typing Hebrew using Latin characters

3.4. TEMPLATES

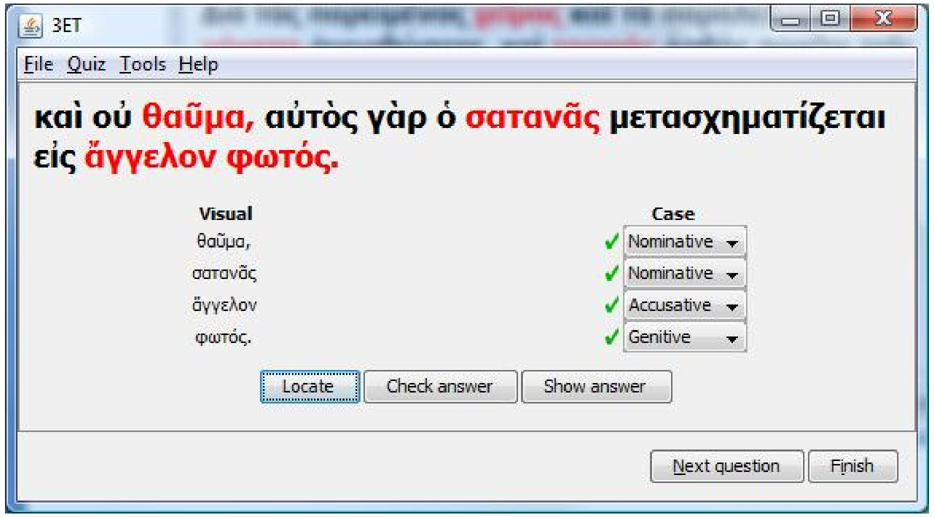
As mentioned in the final paragraph of section 3.1, the specification of an exercise can be saved for later use. Saved exercise specifications are known as *exercise templates.*

To help students run meaningful exercises, a teacher can create a number of useful templates that reflect the course material of a particular language course.

3ET also comes with a number of pre-made templates.

3.5. OTHER CORPORA

A number of different databases are available for 3ET. Figure 14 shows an example of an exercise taken from the Greek New Testament. There is, however, nothing in 3ET that ties it to Biblical texts. Any Emdros database with a fully analyzed corpus can be easily adapted for use with 3ET.



καὶ οὐ θαῦμα, αὐτὸς γὰρ ὁ σατανᾶς μετασχηματίζεται εἰς ἄγγελον φωτός.

Visual		Case
θαῦμα,	✓	Nominative
σατανᾶς	✓	Nominative
ἄγγελον	✓	Accusative
φωτός.	✓	Genitive

Figure 14. Greek exercise in noun cases (2 Corinthians 11:14)

4. GENERATING QUESTIONS FOR MOODLE

Many teaching institutions run Moodle[6]. According to the web site, "Moodle is an Open Source Course Management System... It has become very popular among educators around the world as a tool for creating online dynamic web sites for their students. To work, it needs to be installed on a web server somewhere, either on one of your own computers or one at a web hosting company."[7]

One of the many splendid features of Moodle is its ability to build a bank of questions from which a teacher can create exercises or exams for students.

3ET has an export facility that enables a teacher to generate a number of language questions and export them to the Moodle question bank. Before the questions are transferred to Moodle, the teacher can review the questions and perhaps delete any question that is deemed unsuitable for the purpose.

[6] See http://moodle.org
[7] http://moodle.org/about (retrieved 6 May 2009)

5. STRENGTHS AND WEAKNESSES

Many language teachers create grammar exercises for their students on a regular basis. Often these exercises are "canned" in the sense that they are based on sentences invented by the teacher.

Contrary to this, 3ET draws its exercises from a genuine corpus of real text. This means that the students are exposed to "the real thing", which has both advantages and drawbacks:

Since 3ET chooses the sentences at random, frequent word forms will occur more frequently in exercises than rare word forms. This is good because students primarily need to be familiar with the most frequent word forms; but it is bad because a student may already be familiar with the frequent forms and needs to train the less common word forms. The teacher (or whoever creates the exercise template) can alleviate this problem by specifically excluding common word forms from the sentence selection. For example, when training Hebrew verb forms, the sentence selection may specifically exclude the very common Hebrew verb stem qal as we did in the example in Figure 4.

The student should also be aware that since we are dealing with a real text, the writer may have made grammatical errors.

Finally, some sentences are ambiguous and can therefore be analysed in several different—but equally valid—ways. An Emdros database normally includes only one interpretation of each sentence, which may differ from the way the student reads the sentence.

If 3ET is being used to generate exercise questions for Moodle, the teacher will probably want to remove ambiguous and ungrammatical sentences before exporting them to Moodle.

6. IMPLEMENTATION

Figure 15 shows the architecture of the 3ET application.

3ET itself is written in the Java programming language which gives a number of advantages. Currently 3ET is available as a stand-alone program from PCs running the Windows operating system. However, because of the inherent platform-independence of Java, a web-based version of the application can easily be written, which would make it possible to run the program as a web service rather than as a program that must be installed on a computer.

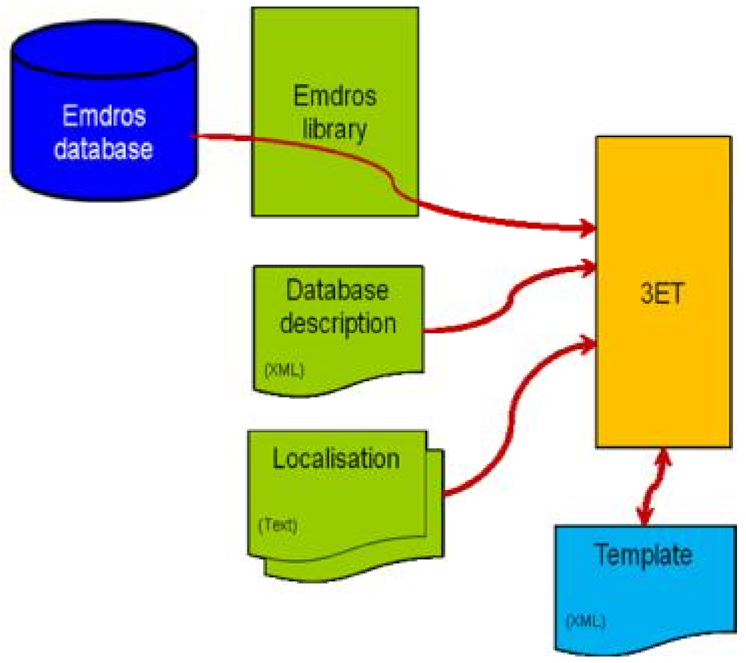


Figure 15. Application architecture

As the illustration shows, 3ET uses a software library to access the relevant Emdros database. Contrary to 3ET proper, this library is not platform-independent and must therefore be tailored to the specific operating system on which it is running. For the Windows version of 3ET, this means that a Windows DLL[8] is part of the distribution.

Although Emdros databases include a description of the features and values of the objects in the database, some additional information is required to make the program user-friendly. This extra information is found in the files "Database description" and "Localisation" in Figure 15.

The database description file is an XML[9] file that lists which objects and features are relevant for the application and which are not. It also identifies the features that must cause 3ET to hide words as shown in Figure 12. Finally the objects that define the text hierarchy (book/chapter/verse for biblical texts) are specified in this file.

[8] Dynamic-Link Library.
[9] Extensible Markup Language.

All the information in the database description file is language independent. The localisation files provide a translation of the object and feature names into the language of the user. Currently, localisation files for English and Danish are available; but a version in some other language can easily be made by creating a few text files. There is one localisation file for the application itself and one file for each Emdros database.

Finally, Figure 15 shows that 3ET reads and store exercise template files as described in section 3.4.

7. AVAILABILITY

At the time of writing (May 2009) 3ET is still a beta-level product. A few adjustments and corrections to the code are still being made. The intention is that 3ET will be delivered with a Greek New Testament text plus a few chapters of the Hebrew Old Testament. A full Old Testament text will be available at an additional cost, but the exact licensing terms are still under discussion.

8. THANKS

I am very grateful to Associate Professor Nicolai Winther-Nielsen of the Copenhagen Lutheran School of Theology for teaching me the basics of Hebrew grammar and for advice and inspiration in the development of 3ET. Thanks also go to the creator of Emdros, Assistant Professor Ulrik Sandborg-Petersen of Aalborg University, for quick and useful help on how to use Emdros.

9. REFERENCED WORKS

Petersen, Ulrik (2004). "Emdros – A Text Database Engine for Analyzed or Annotated Text". In: ACL, COLING 2004 Geneva, 20th International Conference on Computational Linguistics, August 23rd to 27th, 2004. Volume II. Proceedings, pp. 1190-1193. (Available as a PDF file from http://emdros.org/petersen-emdros-COLING-2004.pdf.)

Winther-Nielsen, Nicolai (2009) "Biblical Hebrew parsing on display: The Role-Lexical Module (RLM) as a tool for Role and Reference Grammar." In *Hiphil 6* [http://hiphil.see-j.net] 2009.