A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Computer Science and Engineering

in the Graduate School of the

University of Aizu

Word Bricks: a Virtual Language Lab

by

m5162103

Roman Efimov

Reviewed by

Associate Professor Maxim Mozgovoy, Main Referee

Professor Vitaly Kluev,

Professor Irina Khmyrova

August 2014

The thesis titled

"Word Bricks: a Virtual Language Lab"

by

m5162103

Roman Efimov

is approved:

Main referee
Associate Professor Date
Maxim Mozgovoy

Professor
Vitaly Kluev Date

Professor
Irina Khmyrova Date

University of Aizu

Summer 2014

Table of Contents

Abstract

The use of computer-assisted language learning (CALL) instruments is now widespread. However, popular CALL systems still rarely incorporate modern achievements of natural language processing technologies. One can note a contrast between CALL instruments and educational software, available for physics or chemistry. For these subjects, a student can often perform numerous open experiments in a “virtual lab”. WordBricks aims to implement a similar idea of a “virtual language lab” based on established natural language processing technologies.

We create a system that operates as follows. A student is given a number of “word bricks” that represent single words. The student can connect individual bricks to form phrases and sentences. Every brick has typed incoming and outgoing “connectors”, ensuring that only grammatically correct links are possible. Such a system will enable students to play with word combinations, to test hypotheses about the correctness of certain grammatical constructions, and to receive the system’s feedback.

The system will be based on widely known dependency grammar formalism, representing sentences as trees of directly connected words. At the initial stage of the project, the set of admissible links for every part of speech will be defined manually, but later we plan to derive grammatical rules automatically from a dependency treebank (a manually annotated corpus of sentences represented in form of dependency trees).

In this thesis we describe how this program works, some limitations of using this program, and difficulties we met.

Introduction

When computers became commodities, terms like “computer-assisted X” lost some significant part of their initial meaning. We do not refer to “ballpoint pen-assisted writing” or “car-assisted traveling”, and yet “computer-assisted language learning,” or CALL, is still in common use. In regard to CALL, we should probably imagine dedicated educational systems that somehow “assist” learning in a nontrivial technologically-driven way, but ironically common definitions of CALL simply refer to the use of computers in language learning activities [1], [2]). For example, according to Beatty, CALL is “any process in which a learner uses a computer and, as a result, improves his or her language” (p. 7). In particular, using an electronic dictionary or watching a foreign-language clip on YouTube are perfect examples of “computer-assisted language learning”, though neither an electronic dictionary nor a video-sharing website are dedicated CALL systems in the sense of being explicitly designed to support language learning activities. In a sense, those systems are similar to pencils and notebooks; they are handy in an educational process, but can hardly be considered “language learning tools”.

Furthermore, it also seems to us that such general-purpose software is the most widely used and most helpful for the learners. By contrast, there are hundreds if not thousands of available dedicated software packages for language acquisition, but strikingly they are rarely mentioned in numerous “language learning tips” found online (see, e.g., [3], [4], [5]). These tips are consistent with our own observations of actual classroom practices, where the use of computer technology is typically limited to electronic dictionaries, video-sharing websites, and learning management systems.

In general, computer technology holds a firm position as a helper within traditional teaching and learning practices. We learn language by listening, speaking, reading, writing, and doing (established) exercises, and computers provide unprecedented support and

convenience in these activities. However, overall they still fail to provide fundamentally new teaching and learning practices, unavailable in traditional paper-and-pencil scenarios.

Even dedicated CALL systems (such as the ones developed by companies like Eurotalk, Berlitz or Rosetta Stone) are typically designed as integrated packages of traditional learning materials — audio/video clips, pictures, texts, exercises, and vocabularies. In other words, current CALL systems can be considered primarily as highly usable and modernized versions of traditional “book + tape” self-learning courses. The survey conducted by Hubbard in 2002 revealed that even the CALL experts are not convinced about the effectiveness of educational software. Hubbard notes: “…it is interesting that questions of effectiveness still tend to dominate. In fact, the basic questions of "Is CALL effective?" and "Is it more effective than alternatives?" remain popular even among those who have been centrally involved in the field for an extended period of time.” [6].

These observations lead to a rather regrettable conclusion: we have multi-gigahertz, multi-gigabyte computing equipment, but it is still used mostly as an enhanced version of blackboards, video players, dictionaries, textbooks, and so on. We suggest that the reasons are both technological and psychological: many computer technologies relevant to language learning are indeed not mature enough to be used in practical CALL systems, and our traditional learning habits make it hard to design fundamentally new systems that would utilize the full power of today’s computing hardware.

In this thesis, we will briefly discuss some of these issues and outline several directions for future research in CALL, both technically feasible and psychologically sound. Also we introduce our project named “Word Bricks”, which can be used for studying English grammar by playing with words and combining them into sentences in

a puzzle-like way. We will discuss its expected advantages and drawbacks as well as possible future research directions.

Related work

Let's make a review of existing technologies in CALL. See table below.

Name of technology	Definition	Example affordances for language study
Course management system (CMS)	Server-based application used to present materials and services required for blended or distance learning (such as syllabi, required readings, calendars, etc.). Teachers and students access a CMS over a network through a web browser, using a menu-driven interface	• Enable sharing of course materials, allowing accessibility to content anytime, anywhere • Facilitate course content organization and teacher–student and student– student communication
Interactive white board	An interactive display that comprises three pieces of equipment: a computer, a projector, and a display panel, which is a large freestanding or wall-mounted touch-sensitive screen. The projector displays the image of the computer screen on the screen, which is easily viewable by all students in the classroom	• Promote interactive activities and engage students and teachers in collaborative work • Enhance motivation and improve attitudes toward learning • Incorporate authentic content available on the internet into classroom lessons
ePortfolio	A digital archive of student work created by a learner that records evidence of the learner's experiences, progress, achievements, and self-reflections	• Support learner autonomy and selfassessment • Emphasize the process of learning, rather than just the products of learning • Facilitate setting learning goals, monitoring progress, and developing self-assessment skills [7]
Corpus	A collection of authentic language in spoken form, written form, or both. Corpora vary in terms of design (fixed size vs. expandable), content (general vs. specialized), and medium	• Provide access to rich, authentic input • Enable broad access to linguistic data • Promote data-driven inductive learning [8], [9]

	(written vs. spoken)	
Electronic dictionary	A dictionary in electronic form – either handheld or online	• Speed searches for a lexical item so that looking up words does not greatly interrupt the reading process • Accommodate different look-up preferences and learning styles • Support individualized and elaborated input
Electronic gloss or annotation	A method of reference, usually in a form of a hyperlink, that allows learners to access glosses (word- or sentence-level, context-specific translations) or annotations (explanatory or background information) while reading an electronic text	• Provide for efficient lookup of unknown words and multimedia capability • Facilitate reading comprehension, and incidental and intentional vocabulary learning
Intelligent tutoring system	A program that simulates a tutor by providing direct, customized instruction and/or feedback to a learner. Such a system is generally comprised of four components: an interface (platform), an expert model (domain of knowledge the student is intended to acquire), a student model (current state of student's knowledge), and a tutor model (which provides appropriate feedback and instruction by using the identified gaps between the student and the expert models)	• Tailor instruction to the individual learner • Provide immediate, specific feedback in a systematic manner • Can implement taskbased interfaces in language instruction
Grammar checker	A program designed to evaluate a written text's well-formedness in terms of grammaticality. Such programs are often packaged, along with spellcheckers, within word-processing programs	• Identify/flag low-level morphosyntactic errors [10], [11] • Provide students with immediate input and feedback

Automatic speech recognition (ASR) and pronunciation program	A technology that allows a computer to identify the words a person speaks into a microphone. ASR is often a component of speech pronunciation software, and as such, identifies particular parameters of the learner's output, such as prosody or specific sounds, and provides feedback on these aspects of performance	• Compare student's pronunciation acoustically with a target pronunciation and provide feedback • Provide learner with an opportunity to work on speaking ability individually, at selfselected pace • Allow learner to practice simulated dialogue with computerized agent
Virtual world or serious game	A virtual world is a program that allows learners to move a representation of a character, or ''avatar'', through a 3-D graphical environment. A serious game is a virtual environment or traditional computer game in which activities are guided or restricted by the program and users have a specified goal or set of goals to complete	• Provide virtual meeting spaces • Enable learner to navigate within simulated environments, including those modeled after target language locales and incorporating culturally relevant objects • Encourage role play through the ability to embody different characters within a scenario
Chat	A form of synchronous computer-mediated communication; either textbased or include audio	• Record logs of interactions, which can be printed for review and used as an assessment tool [12] • Enable communication and collaboration among students or between students and native speakers without constraints of distance or location
Social networking	Social networking, of which Facebook and MySpace are the best-known examples, enables peer-to-peer communication and collaboration. Users develop their own presence on social networking by creating profile pages about themselves, and	• Support networking among users as well as the ability to communicate with others with similar interests • Enable interaction with native speakers and other students of the target language • Allow for synchronous and

	then joining networks based on geography, interests, associations, or friendships	asynchronous communication
Blog	A web application that displays entries authored by the blog owner with time and date stamps and is visible to other web users	• Support personal journaling or blogging and enable feedback in the form of comments on blog posts E • Encourage collaborative learning
Internet forum or message board	An asynchronous system in which messages are sent to multiple recipients. Messages are threaded according to topic and a notification is often sent to a user's e-mail address when an update is posted	• Organize discussions via topic thread • Enable online information exchange without constraints of time and distance
Wiki	A website that allows multiple users to post or edit information	• Help students and instructors to find information easily through organization by topic • Enable collaboration on in-class projects • Support open-editing of content

Table 1. List of technologies in CALL.

Although the use of technology to enhance FL learning and teaching has grown rapidly during the past three decades, most research has focused on its viability for supporting FL learning; very few well-designed empirical studies support its efficacy for improving FL learning processes or outcomes. Rather, most CALL studies seem to focus on either describing the affordances offered by particular types of technology or measuring their effects on students' affective reactions, such as increased motivation or increased enjoyment of learning activities. A large number of studies confirmed that learners enjoy using technology in FL learning and that they prefer using technology over more traditional methods and materials. Because of technology, learners tend to be more engaged in the process of learning, and have a more positive attitude towards learning [13].

But there are a small number of technologies teaching grammar. There are just intelligent tutoring system and grammar checker. Also you can see in this review, there is no “open labs” like software. Thus a situation we have open labs in physics, chemistry and other areas but don’t have it in CALL. And this lack of software motivated us to develop Word Bricks.

CALL Meets Technological Limits

Educational language learning software that can do more than merely support traditional learning activities is known as Intelligent CALL (ICALL) systems. The role of ICALL can be also viewed as being "teacher's assistants" that assume some of the teacher's basic teaching functions. Presently, there are three main directions in ICALL research: (1) identification of errors in student input and generation of necessary feedback, (2) adaptive knowledge and exercise delivery based on a student's current proficiency, and (3) creation of automated conversations agents (chatbots). However, the overall capabilities of present day ICALL systems are limited.

We can add that research efforts in this area are limited, too. For example, Volodina et al. observe that only three natural language processing-backed CALL systems have come into everyday classroom use [14]. (All three systems evaluate student input and provide interactive feedback.) Furthermore, as noted in [15], "the development of systems using NLP technology is not on the agenda of most CALL experts, and interdisciplinary research projects integrating computational linguists and foreign language teachers remain very rare".

This observation is indirectly supported by Hubbard, who identifies the following three areas for the current development of CALL systems (and none of them includes ICALL): Web 2.0 (collaboration, social networking, blogs, video publishing platforms, etc.); mobile platforms (as an opportunity for ubiquitous access to learning materials); virtual worlds (as a way to bring together people into a shared virtual space) [16].

Possibly, the only "intelligent" technology that has made its way into some retail CALL systems is automated speech analysis, which is used to evaluate the quality of student pronunciation. Such an instrument is implemented, e.g., in commercial Rosetta Stone software, but its resulting quality is sometimes criticized [17].

We have to state that future development of ICALL systems crucially depends on significant achievements in the underlying technologies. Language learning is a sensitive area, where misleading computer-generated feedback may harm students. So it is impossible to expect any rise of ICALL systems before the related natural language processing technologies improve vastly.

Virtual Labs

However, by contrast, such a technology-backed, student-centered approach is already implemented in a number of educational systems for the disciplines such as physics, chemistry, and computer science. Notably, there are sandbox-like environments (or “virtual labs”) that do not restrict their users and do support open experimentation.

For example, Open Source Physics project [18] collects together a vast amount of interactive physical simulations with user-adjustable parameters. The 2D physics sandbox Algodoo is positioned by its authors as *“the perfect tool for learning, exploring, experimenting and laborating* [sic] *with real physics”* [19].

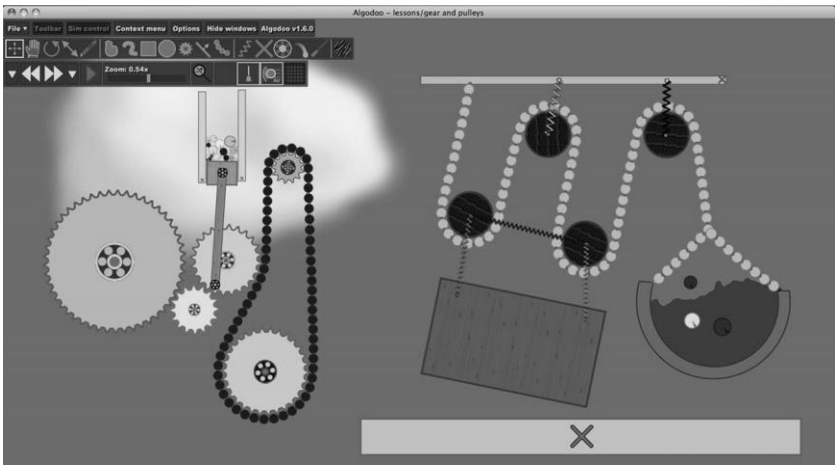

Fig 1. Screenshot of Algodoo.

The ChemCollective collection [20] includes a number of ready setups for chemical experiments as well as a virtual lab for open exploration.

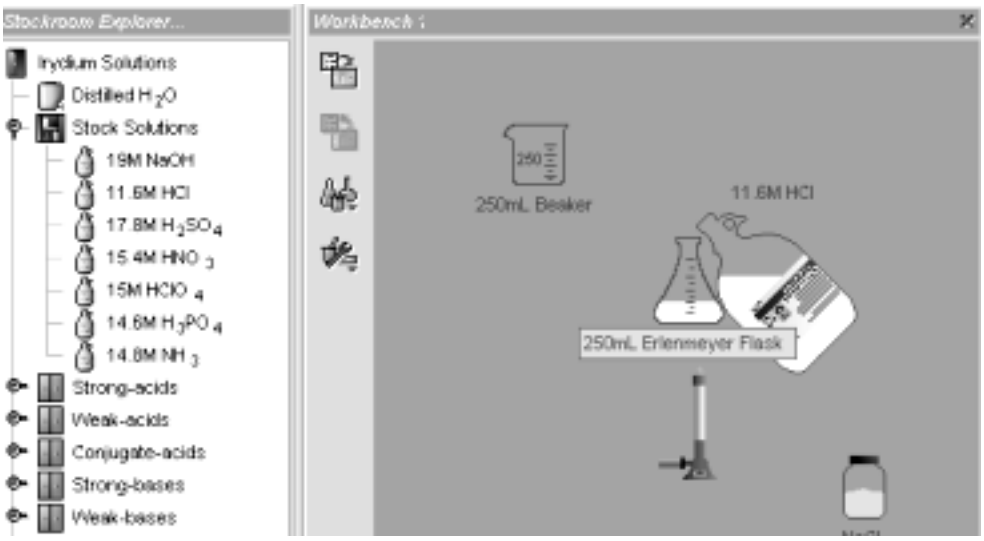

Fig 2. Screenshot of Chemcollective.

The JFLAP environment [21] allows students to create, analyze and test finite-state machines — the devices that constitute the basis of computer science.

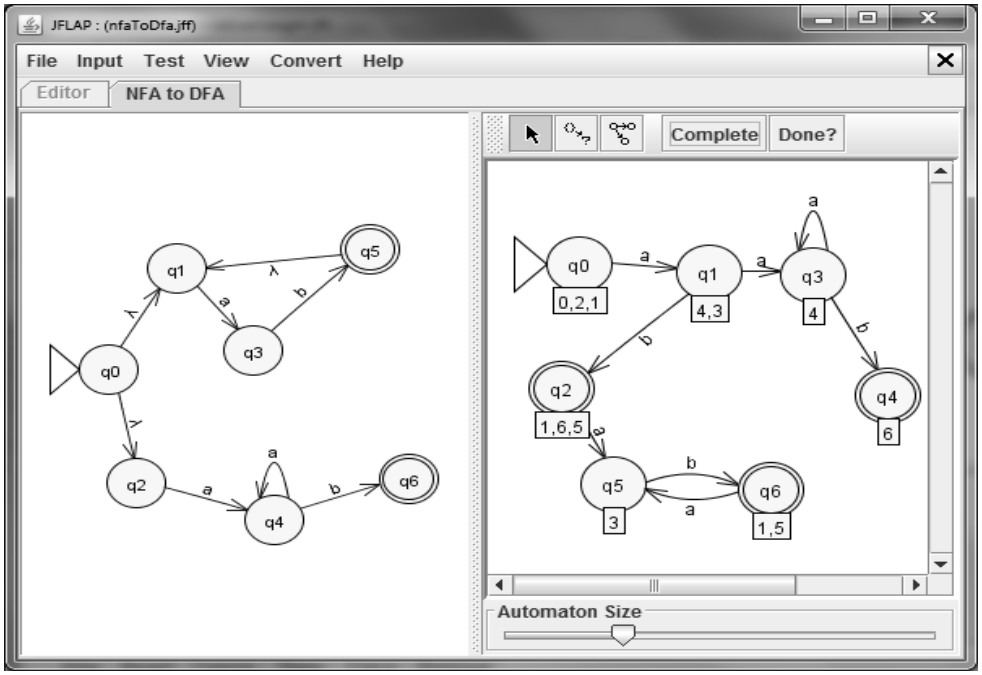

Fig 3. Screenshot of JFLAP.

We consider such systems as great examples of well-grounded uses of computer technology in education. Virtual labs provide safe and controlled environments in which students can test their ideas, and in

this sense they can be likened to flight simulation software, used to train pilots: the students perform predefined training routines, but also can experience the outcome of any arbitrary maneuver. Furthermore, virtual labs contribute to the modeling of the problem domain in the learner's mind, and thus are consistent with constructivist views on educational process.

It is interesting to note that from the technological point of view, virtual labs are not necessarily complex systems. The possibility of open experimentation outweighs many technical limitations and constraints. Say, for the physical sandbox example, even a simple system that allows its users to examine the collision of balls of different speeds and masses in a two-dimensional space may have educational value.

Unfortunately, environments for open experiments are barely provided by the existing CALL systems. This perhaps can be attributed to the unclarity of the notion of an "experiment" in language learning. It is evident, however, that a large portion of active language learning is related to the process of combining words and phrases into meaningful sentences, and the analysis of the subsequent feedback. We learn a language both by comprehending other people's speech and writing, and by creating our own phrases that are to be tested for admissibility by our interlocutors.

Within such a concept of experiments, even a feature-rich electronic dictionary can be a powerful experimental tool in the hands of an avid learner. Indeed, with full-text search it is possible to check actual word use, test the correctness of certain word combinations, the compatibility of certain prefixes with certain stems, and so on.

The ways in which language learners could do "experiments with the language" are still to be identified. Here we can only quickly introduce our own work-in-progress system that is intended to help language learners master basic grammatical rules.

The Basic Concept of "Word Bricks"

We decided to devote our project to one specific type of language learning activities: to the process of constructing grammatically correct phrases. A student with initial vocabulary and some knowledge of grammar rules might want to practice them by creating simple sentences. At this stage, it is important to make sure that the sentences are built properly, and if not, the student gets necessary feedback. By creating sentences, the student in the simplest case can test hypotheses about the correctness of certain constructions. In more advanced scenario, the feedback might include hints on the proper use of words and word combinations. For example:

- A student can check whether a certain word is appropriate in a certain context. Suppose the student knows that one can *ride a horse*, but can one *ride a car*?

- A student can find the correct word form for the given syntactical context. In English, the verb form depends on the subject's person, so the student has to choose between the base form of the verb and the 3rd person singular form. For other languages these rules can be more complicated. For example, Russian verbs are conjugated according to the subject's person and number in the present tense, but to the subject's gender and number in the past tense.

- A student can find correct prepositions and/or grammatical cases for the given context. For example, in Finnish some verbs require that the object noun is always set into a certain form (so the verb "governs" the noun). This verb / noun form list has to be memorized.

The idea of incorporating a grammar checker into CALL software is not new. Such an automated feedback generation system was implemented, e.g., in Robo-Sensei Japanese tutoring system [22]. However, today's grammar checkers are not very helpful in open

experiments with language constructions. As noted in [23], grammar checkers are usually aimed at native speakers, and do not provide sufficient feedback for language learners. One possible way to solve this problem is to restrict user input. This approach is implemented in Robo-Sensei: the system asks the student to answer a specific question, and then compares the response with an “answer schema” that specifies the pattern of the expected correct response.

We believe that free experiments with language constructions are possible without traditional grammar checking technologies. Consider the following analogy. A programmer, working with traditional programming languages, has to write plaintext code that is translated into low-level machine instructions. It is a job of a compiler or interpreter to parse the code, and to identify possible syntactic errors. Unlike them, *visual programming systems*, often used for teaching programming to kids, store programs in graphical flowcharts (see, e.g., Flowol [24]), thus eliminating the need of parsing and error checking. One can draw a flowchart that corresponds to a wrong algorithm, but the flowchart itself cannot be “syntactically incorrect”, since the visual editor allows no illegal links between the elements.

In a sense, flowcharts represent “parsed” programs, stored in the form that directly reflects their syntactic and semantic structure. Natural language sentences also can be represented in a parsed tree-like form with *phrase-structure* grammars or *dependency grammars* [25]. Our idea is to let the students compose *parsed* sentences directly instead of traditional writing.

System architecture and software basics.

This program was writing on Python 3.4 using graphical library PyQt. We used Python Tools for Visual Studio.
Python was choosing because of its dynamical structure. It allows real-time to describe the dynamic structures - bricks.

Each brick is an object containing the list of child objects, which can be either brick or other connector (placeholder). So simple brick can be part of the n-tree, which corresponds to a phrase or part of a phrase.

The only algorithm used here is an algorithm which determines whether or not to connect the connector and the bricks. It compares the shape, connectors and grammatical icons. And according allows connection.

There are three main files in the project: Bricks.py, WordBricks2.py and Vocabulary.py.

Bricks.py contains two main classes: Brick and BrickConnector.

WordBricks2.py contains main procedure which runs the program.

Vocabulary.py contains the list of words which used in the program.

```
def dog():
    det = BrickConnector(Brick(TypeDeterminer(), [IconSingular], [], []), True, True)
    prop = BrickConnector(Brick(TypeProperty(), [], [], [[WordPlaceholder()]]), False, True)
    return Brick(TypeObject(), [IconThirdPerson, IconSingular], [AnchorSubject(), AnchorObject()], [[det], [prop], [WordPlaceholder("dog")]], "dog")

def dogs():
    det = BrickConnector(Brick(TypeDeterminer(), [IconPlural], [], []), True, True)
    prop = BrickConnector(Brick(TypeProperty(), [], [], [[WordPlaceholder()]]), False, True)
    return Brick(TypeObject(), [IconThirdPerson, IconPlural], [AnchorSubject(), AnchorObject()], [[det], [prop], [WordPlaceholder("dogs")]], "dog")

def I():
    return Brick(TypeObject(), [IconFirstPerson, IconSingular], [AnchorSubject()], [[WordPlaceholder("I")]], "I")

def her():
    return Brick(TypeObject(), [IconThirdPerson, IconSingular], [AnchorObject()], [[WordPlaceholder("her")]], "her")
```

Fig. 4. Fragment of Vocabulary.py

In the future, we intend to split the file into several different units. Each unit may contain its own set of words and meet eg an individual lesson.

General Design of the System

The idea of modeling syntactic rules with shaped bricks was implemented in the successful educational programming environment Scratch [26]. In Scratch, individual syntactic elements of a computer program, such as control flow statements, arithmetic expressions, and logical operations are represented with shaped bricks that have to be combined to constitute a program (Figure 1). While Scratch programs may have logical errors, syntactically they are always correct, since it is impossible to combine mismatching bricks.

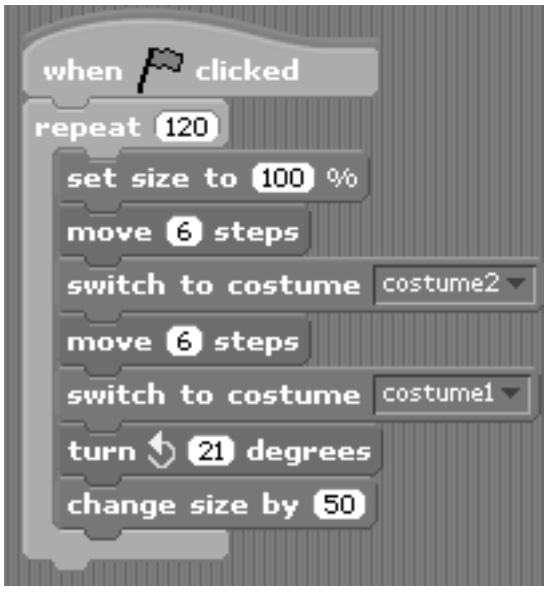

Fig 5. A fragment of Scratch program.

As such, Scratch's graphical editor is not just a simpler way to write computer programs, helpful for the beginners.

In our research, we are working towards implementation of a similar scheme for natural language sentences. Undoubtedly, natural language grammar is much more complex and less formal than the syntax of any programming language. However, for the purposes of novice language learners, it is reasonable to teach restricted grammar (as it happens in traditional language teaching), and this is technologically feasible.

Even in the case of Scratch, the design of brick linkage principles is not trivial. One important problem is to make sure that the links between the bricks reflect *actual* structure of the corresponding computer program. For example, a loop control structure can be theoretically represented with the separate “Begin Loop” and “End Loop” bricks that surround bricks that constitute the loop body; however, such a design would make a false impression that “Begin Loop” and “End Loop” are independent program elements. Instead, a loop in Scratch is represented with a single C-shaped brick that embraces the loop body.

It is much harder to identify a consistent set of rules that control such linking principles of a natural language-based system. However, they are actually considered in a number of linguistic theories. In particular, we base our rules on the principles of *dependency grammars* [27]. Existing guidelines, such as the Stanford Typed Dependencies Manual [28] describe in detail how the words in the given sentence should be linked to form a structure consistent with the ideology of dependency grammars. For example, a subject and an object should be directly connected to their head verb; an adjective should be directly connected to its head noun (Figure 5).

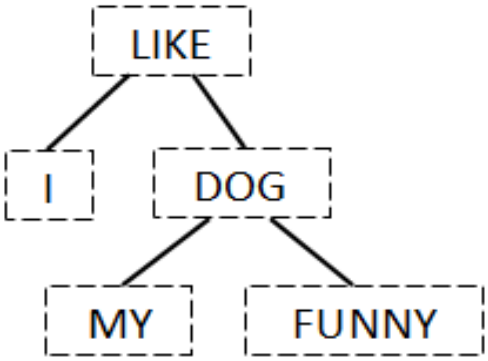

Fig 6. Dependency tree of the phrase “I like my funny dog.”

The resulting structure of a sentence is represented with an *n*-ary tree. While this structure is linguistically correct (according to the theory of dependency grammars), it arguably might be difficult for

learners to master it. Therefore, it is our challenge to represent such trees as two-dimensional brick puzzles. Furthermore, dependency grammars do not express word order, while it has to be reflected in the resulting brick structure (Figure 7).

Fig 7. Dependency tree of the phrase “I like my funny dog” in the form of 2D puzzle.

The proposed learning environment can be used in different scenarios, but we would emphasize again the possibility to perform open experiments. Learners will be able to test which word combinations are admissible and why.

We should also note that it is an open question whether language learners (at least in the early stages of learning) should study sentence structure. However, we believe that some gentle exposure is fruitful, especially for learning languages with rich morphology, where a single change in one word may trigger changes in several of its dependent words.

My contribution to the project

My main purpose in the Word Bricks project was development of the rules of grammar visualization and consistency check. Also I conducted experiments with students.

First steps in the development of the visualization were difficult. We had dependency tree but not to know how visualize it. There is a lot of word's attributes. Words can be different types, for example nouns or pronouns. It has different grammar cases (can be objects or subjects in a sentence). And finally it has shape and number (1st, 2nd or 3rd shape, plural or singular number). And all these attributes should be visualized.

As mentioned earlier, the good example of the visualization of sentence is "children's" programming language. Take an example of «Scratch».

Program in «Scratch» is logically correct vertical sequence of blocks of different shapes and colors that connect to each other with connectors. The shape and color of the block is determined by its purpose. Connectors are as different from each other in shape (like puzzles). Therefore, put the block in the wrong position is obviously impossible.

Of course, the syntax of a programming language is easier than natural one. In first case most of the typical items go one after the other and there is small number of connectors needed. For example, you can connect two logically different operations syntactically suitable to each other. In case of natural language we have more dependencies. Therefore connectors should be more.

So we decide to use Scratch rules in our research. Below you will see first two versions in visualization grammar (Figure 8).

It's not clear in this picture, but every word has different color. Also every word has different shape. In the first version (see on the left

side of the picture) there are no connectors. In the second version (see on the right side of the picture) we have it.

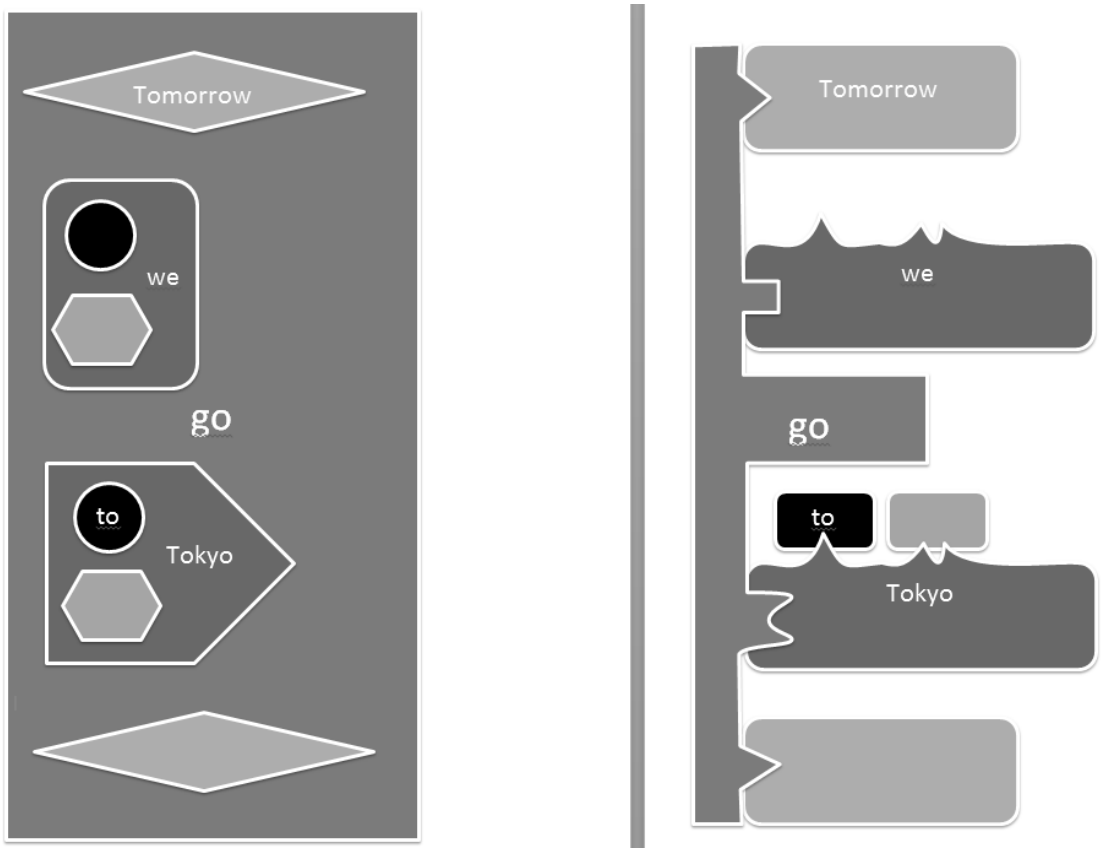

Figure 8. Two versions of visualization grammar.

About modern view of Word Bricks we told you in the next chapter.

Word Bricks: How It Works

In this chapter we briefly explain how Word Bricks works. We also show examples and screens of working application.

As you see in the Figure 9, the working area divided into three regions: Words Bank, Word Forms and Constructing Area.

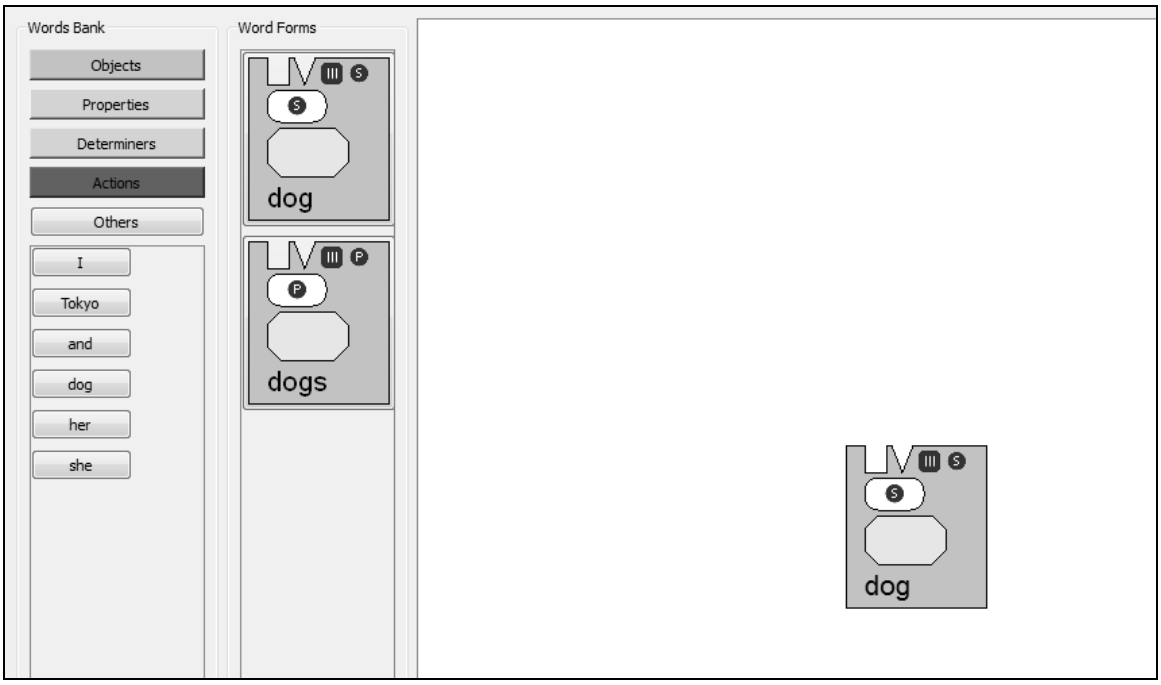

Figure 9. Worksheet

At this time in Words Bank there are 5 groups of words:

- Objects - nouns and pronouns.
- Properties – adjectives and conjunction “and”.
- Determiners – prepositions (like “a”, “my”, “the”).
- Actions – verbs.
- Others – prepositions “from” and “to” and keywords of time (“tomorrow”, “yesterday” etc.)

When you choose a word from Word Bank, you can see in Word Forms different forms of this word, for example “dog” consists of singular and plural forms of it: “dog” and “dogs” (see in Figure 9).

Every word has 3 unique markers (Figure 10): shape, connectors and grammatical icons.

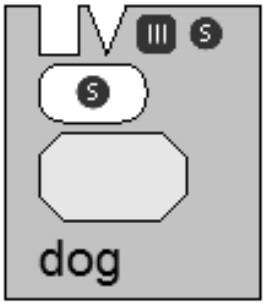

Figure 10. Word Brick

- Shape and color represents type of word (noun, verb, adjective etc.).
- Connectors are used to indicate the word grammar case (for example can it be subject or object).
- Icons represents word’s attributes (shape and number).

As an example, choose word “dog” (Figure 10).
Square shape represents noun.
At the top you can see 2 connectors of square and triangle shapes. It shows that this word can be subject or object.
And finally there are two icons that mean that this word is third form, singular.

Every word can be connected to each other just if all the conditions are the same. For example, we construct a phrase “She likes my dog”. We connect “my” to “dog”, connect “she” and “dog” to “likes” smoothly (Figure 11) because of equivalent of all 3 markers of each word.

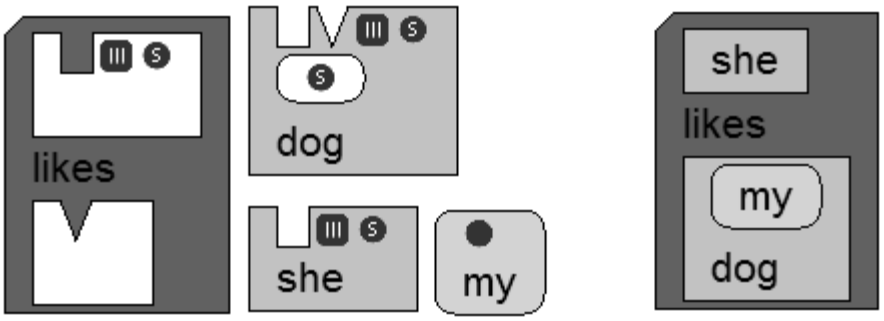

Figure 11. Creation of the sentence.

Next bricks we look at are words-adaptors “from” and “to” and conjunction “and ”.

We have three “and”. One for objects, one for subjects and one for adjectives. This decision has a good reason – for example user cannot combine object with subject (Figure 12).

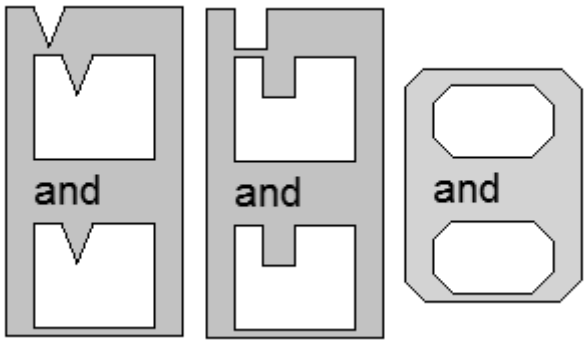

Figure 12. Bricks “and”.

Let’s look at the adaptors “from” and “to”. We cannot create the phrase “I will travel Tokyo”, because the correct phrase is “I will travel **to** Tokyo”. And first we should combine “to” and “Tokyo” to change the case of the word, and then combine this word with “will travel” (Figure 13).

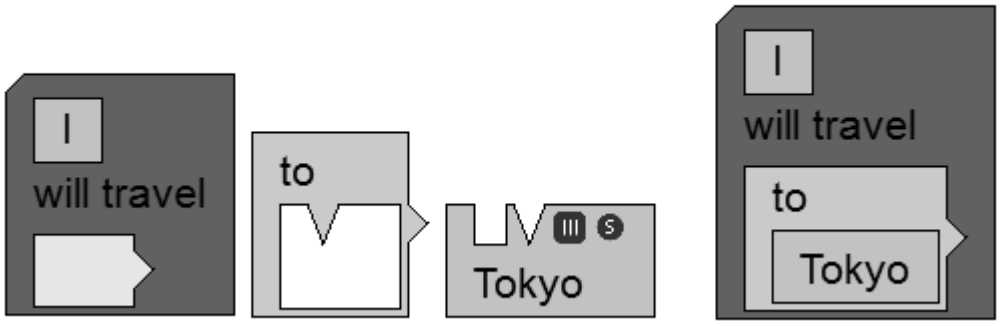

Figure 13. Application of “to”.

Testing and students feedback

Of course our program is not ready right now. There are a lot of steps to complete the project. Anyway we decided to make some tests. We asked 20 students to work with program using simple manual. Then we give them two tasks to create sentences from a given set of words. In the first task students may use just Word Bricks, in the second they may use any tools like e-dictionaries, on-line dictionaries, web systems etc. After that they were given a questionnaire of ten questions including two suggestions: how to improve the graphical interface and how to improve system scenario. Students are very responsibly approached to the matter and gave some interesting recommendations.

Now let's talk about other questions and statistic of students' responces. We show the results in the diagram view.

1. Did you complete Task 1 succesfully?

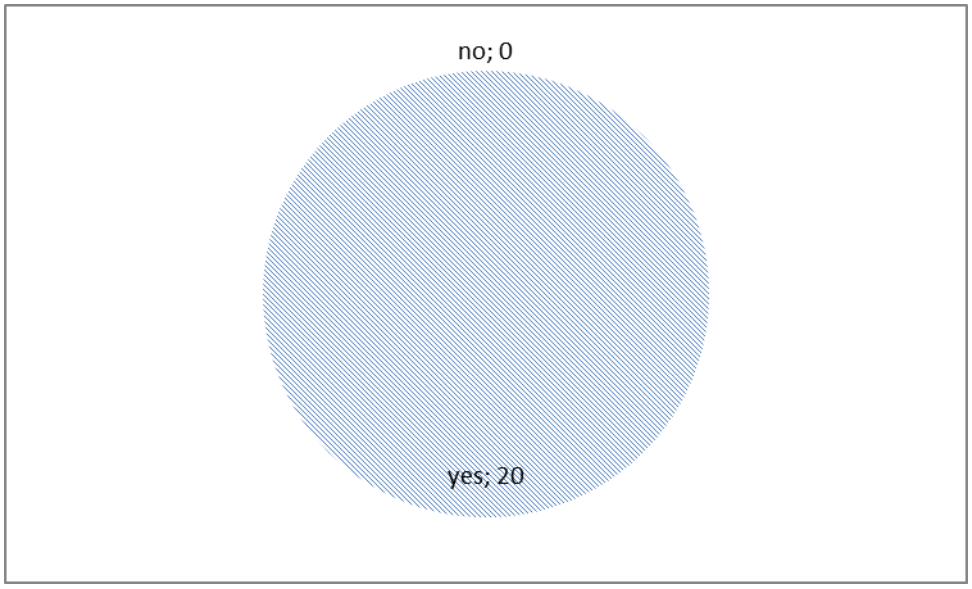

Figure 14. Question 1.

2. What tool did you use while working on Task 2?
   On this question only one student wrote, that he used on-line dictionary. Others wrote that their used nothing.

3. Where Task 1 and Task 2 difficult to you?

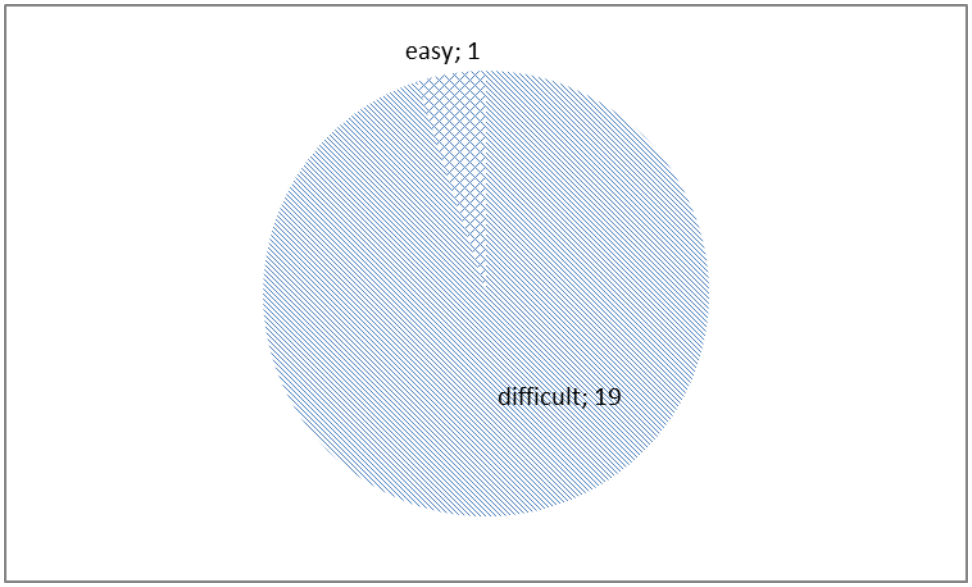

Figure 15. Question 3.

4. Does Word Bricks help you to complete Task 1?

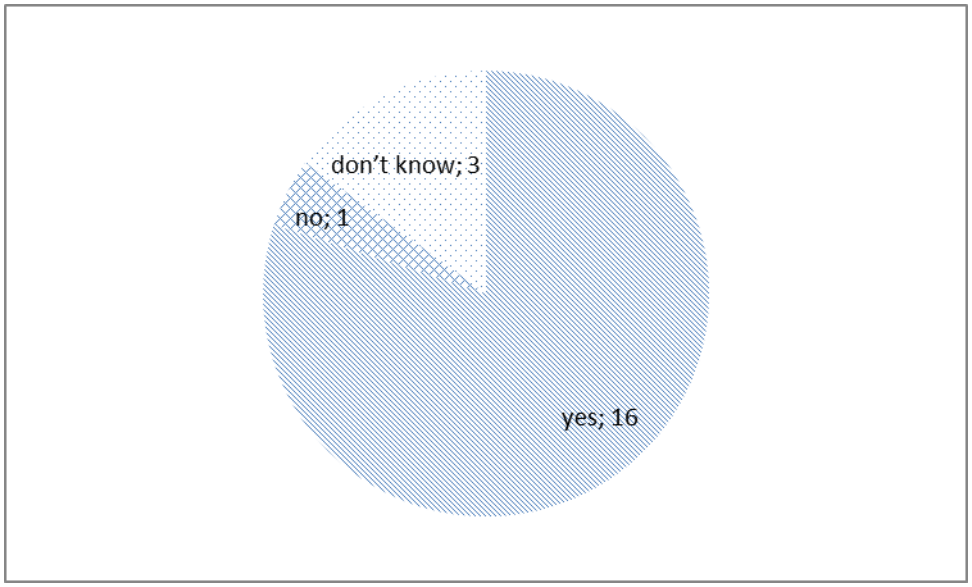

Figure 16. Question 4.

5. Do you like graphical interface of Word Bricks?

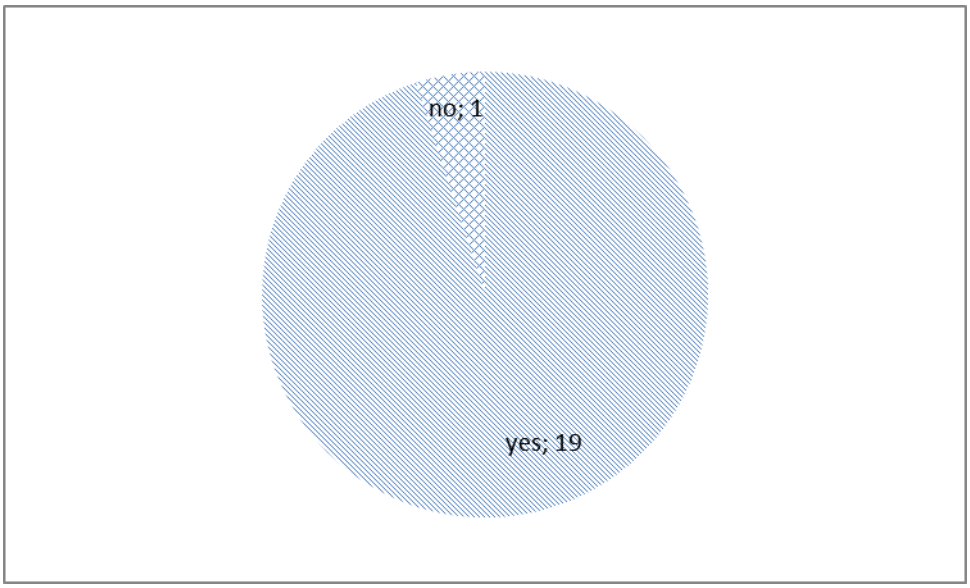

Figure 17. Question 5.

6. Does work with Word Bricks easier than with your tool?

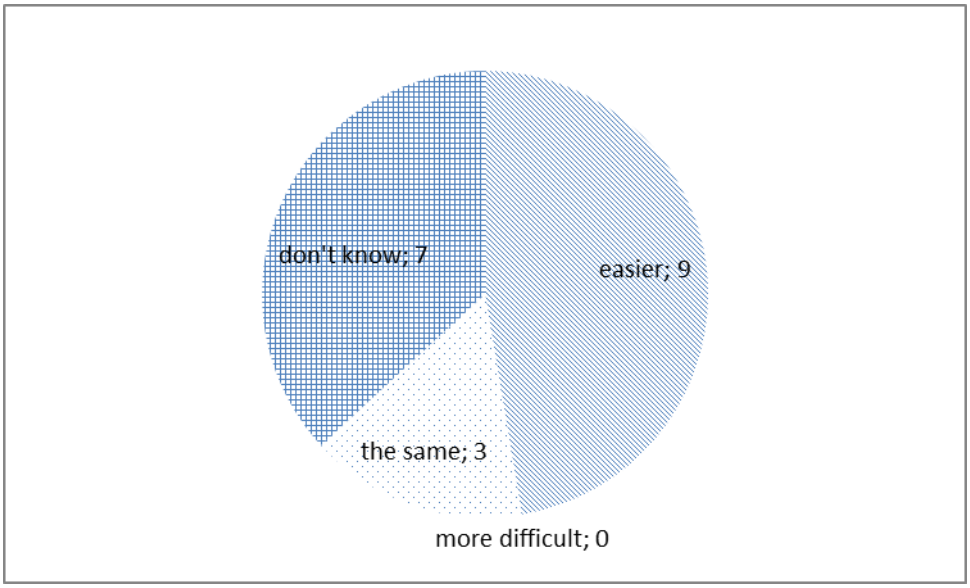

Figure 18. Question 6.

7. How do you evaluate Word Bircks on a scale of 1 to 5 (5 is maximum)?

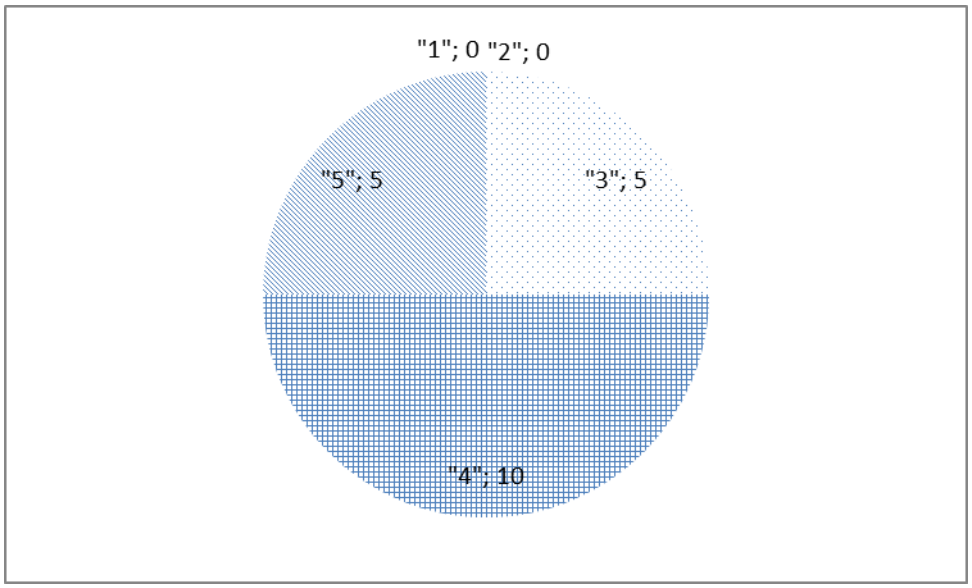

Figure 19. Question 7.

8. How do you evaluate your favorite tool used in these Tasks on a scale of 1 to 5 (5 is maximum)?

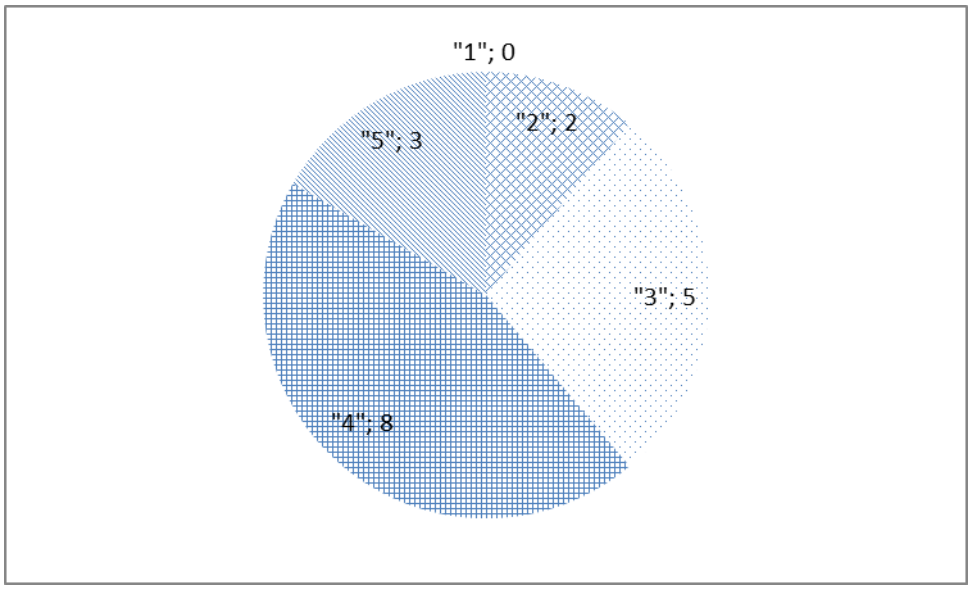

Figure 20. Question 8.

Based on detailed analysis of students' responces, all the students complete all the tasks succesfully. Just one used another tool (on-line dictionary) to complete the tasks. And in general, students estimate the project positively.

Discussion

Let’s talk about Word Bricks limitations and second language acquisition theories.

Word Bricks is a formal system. It is not suitable for theories that do not provide a formalized teaching grammar. For example sociocultural theories, focused on spoken communication between people. Such theories like Sociocultural Theory of Lev Vygotsky and Input Hypothesis of Stephen Krashen are not supported Word Bricks. Sociocultural theory of Vygotsky is the notion that human mental function is from participating cultural mediation integrated into social activities [29]. The input hypothesis, also known as the monitor model, is a group of five hypotheses of second-language acquisition developed by the linguist Stephen Krashen in the 1970s and 1980s. The hypotheses put primary importance on the comprehensible input (CI) that language learners are exposed to. Understanding spoken and written language input is seen as the only mechanism that results in the increase of underlying linguistic competence, and language output is not seen as having any effect on learners' ability. Furthermore, Krashen claimed that linguistic competence is only advanced when language is subconsciously acquired, and that conscious learning cannot be used as a source of spontaneous language production. Finally, learning is seen to be heavily dependent on the mood of the learner, with learning being impaired if the learner is under stress or does not want to learn the language [30].

Word Bricks does not fit all students. Some students learn grammar easier in everyday practice.

Also the program is not compatible with grammar books, because grammar is not as clear and simple as WordBricks. Grammar is more complex and contains many exceptions to the rules. It is quite difficult to move it into the Bricks system.

The basic theory on which our project is built is Dependency Grammar.

Dependency grammar (DG) is a class of modern syntactic theories that are all based on the dependency relation and that can be traced back primarily to the work of Lucien Tesnière. The dependency relation views the (finite) verb as the structural center of all clause structure. All other syntactic units (e.g. words) are either directly or indirectly dependent on the verb. DGs are distinct from phrase structure grammars (constituency grammars), since DGs lack phrasal nodes - although they acknowledge phrases. Structure is determined by the relation between a word (a head) and its dependents.

Conclusion

Computer technologies are widespread in modern language education. Some directions in CALL research, such as intelligent systems, have not yet been as fruitful as anticipated, while other developments, such as multimedia and networking capabilities, have surpassed our expectations.

It seems that the present agenda of CALL research is primarily focused on exploring recent technologies such as ubiquitous computing or Web 2.0. However, we see that even basic language learning tools, such as electronic dictionaries or flashcard software, would benefit from greater attention by CALL developers. But such projects as Word Bricks are not so popular in computer linguistics. So our lab tries to develop it and hope that it becomes popular, like open labs in other areas. Yes, we have some difficulties such as complexity of bricks and now in process of decision of this problem.

References

[1] Levy, M. (1997) Computer-assisted language learning: Context and conceptualization. Clarendon Press, Oxford [u.a.].

[2] Beatty, K. (2010) Teaching and researching computer-assisted language learning, 2nd ed. Longman, Harlow, England, New York.

[3] Leick, V. (2013) Tips on managing your language learning, University of Birmingham. http://www.birmingham.ac.uk/facilities/cml/learnersupport/skills/managing.aspx.

[4] Hessian, J. (2012) Tips for Studying Foreign Languages, University of Illinois at Chicago. http://www.uic.edu/depts/ace/foreign_languages.shtml.

[5] Powell, J. (2013) Foreign Languages & Literature: Tips for Studying, Southern Illinois University. http://www.siue.edu/artsandsciences/fll/fll-tips.shtml.

[6] Hubbard, P. (2002) Survey of unanswered questions in Computer Assisted Language Learning, Stanford University. http://www.stanford.edu/~efs/callsurvey/index.html.

[7] Blackburn, J.L., & Hakel, M.D. (2006) Enhancing self-regulation and goal orientation with ePortfolios. In A. Jafari & C. Kauman (Eds.), Handbook of on ePortfolios (pp. 83–89).Hershey, PA: Idea Group Reference.

[8] Bernardini, S. (2004) Corpora in the classroom: An overview and some reflections on future developments. In J. Sinclair (Ed.), How to use corpora in language teaching (pp. 15–38). Philadelphia: J. Benjamins.

[9] Leech, G. (1997) Teaching and language corpora. London: Longman.

[10] Burston, J. (2001) Computer-based grammar checker and self-monitoring. CALICO Journal, 18, 499–515.

[11] Jacobs, D., & Rogers, C. (1999) Treacherous allies: Foreign language grammar checkers. CALICO Journal, 16, 509–531.

[12] Tudini, V. (2003) Using native speakers in chat. Language Learning & Technology, 7, 141–159.

[13] Ewa M. Golonka , Anita R. Bowles , Victor M. Frank , Dorna L. Richardson & Suzanne Freynik (2014) Technologies for foreign language learning: a review of technology types and their effectiveness, Computer Assisted Language Learning, 27:1, 70-105.

[14] Volodina, E., Borin, L., Loftsson, H., Arnbjörnsdóttir, B. & Leifsson, G. Ö. (2012) Waste not, want not: Towards a system architecture for ICALL based on NLP component re-use. In: Workshop on NLP in Computer-Assisted Language Learning. Proceedings of the SLTC 2012 workshop on NLP for CALL. Linköping Electronic Conference Proceedings, 47–58.

[15] Amaral, L., Meurers, D. & Ziai, R. (2011) Analyzing learner language: towards a flexible natural language processing architecture for intelligent language tutors. Computer Assisted Language Learning 24 (1), 1–16.

[16] Hubbard, P. (2009) A General Introduction to Computer Assisted Language Learning. In: Hubbard, P. (ed.) Computer-Assisted Language Learning (Critical Concepts in Linguistics), New York: Routledge, pp. 1–20.

[17] Santos, V. (2011) Rosetta Stone Portuguese (Brazil) levels 1, 2, & 3 Personal Edition Version 4 (TOTALe). *Calico Journal* 29 (1), 177–194.

[18] Christian, W., Belloni, M. & Brown, D. et al. (2013) *Open Source Physics*, http://www.opensourcephysics.org.

[19] Algoryx (2013) Algodoo: 2D Physics sandbox, http://www.algodoo.com.

[20] Yaron, D., Ashe, C., Karabinos, M., Williams, K. & Ju, L. (2013) ChemCollective, http://www.chemcollective.org.

[21] Rodger, S. (2013) JFLAP, http://www.jflap.org.

[22] N. Nagata (2009) Robo-Sensei's NLP-Based Error Detection and Feedback Generation, *Calico Journal*, vol. 26(3), pp. 562-579.

[23] L. Amaral and D. Meurers (2011) On Using Intelligent Computer-Assisted Language Learning in Real-Life Foreign Language Teaching and Learning, *ReCALL*, vol. 23(1), pp. 4-24.

[24] Flowol. URL: http://www.flowol.com

[25] O. Rambow and A. Joshi (1997) "A Formal Look at Dependency Grammars and Phrase-Structure Grammars, with Special Consideration of Word-Order Phenomena," Recent Trends in Meaning-Text Theory, vol. 39, pp. 167-190.

[26] Resnick, M., Silverman, B. & Kafai, Y. et al. (2009) Scratch. Communications of the ACM 52 (11), 60.

[27] Debusmann, R. (2000) An introduction to dependency grammar. Hausarbeit fur das Hauptseminar Dependenzgrammatik SoSe 99, 1–16.

[28] Marneffe, M.-C. de & Manning, C. D. (2008) Stanford typed dependencies manual. Stanford University.

[29] Lantolf, J. P., & Beckett, T. G. (2009) Sociocultural theory and second language acquisition. Language Teaching, 42(4), 459-475.

[30] Nation, P. (2001) Learning vocabulary in another language. Cambridge: Cambridge University Press.