

Example of Using Narratives in Teaching Programming: Roles of Variables

Olga Timcenko

Aalborg University, Copenhagen Institute of Technology,
Lautrupvang 15, 2750 Ballerup, Denmark
ot@media.aau.dk

Abstract. This paper describes a case study of using narratives to motivate non-technology inclined children, 11-15 years old, to learn programming, using LEGO Mindstorms robots and RoboLab graphical programming language. Case study was done during 2004/2005 and 2005/2006 school years, following two different school teams participating in FIRST LEGO League competitions. Using narrative concept and a concept of roles of variables, it was possible to explain several searching and sorting algorithms to children, including an algorithm of finding minimal/maximal value from the set of input values. Results are encouraging and could be generalized to other programming languages but RoboLab.

Introduction

There are successful examples where elements of narratives could be used to teach small children (4-8 years) basis of programming and logical thinking about abstract rules, like ToonTalk [13], and Magical Forest [14], Playground project [15] or LOGO-based microworlds, like Imagine LOGO [16]. In all these environments a child is placed in a fairytale-like virtual environment, and has to invent or solve some tasks, using logical thinking and set of activities that are relevant for developing programming skills.

It is much harder to find evidences of successful applications of narratives in teaching programming skills to college and undergraduate students. However, in papers [11] and [2] two of these trials are described. Waraich [11] uses a multimedia narrative environment to motivate students, especially those who do not have a major in computer science, to learn basis about computer architecture, namely binary arithmetic and logic gates, topics that lots of students find “dry” and “not very interesting”. Andersen et al. [2], describe Lingoland - a Macromedia Director based, adventure-like environment, used to teach liberal arts students programming in Lingo, Director’s language for programming multimedia environments.

Andersen et al. correctly point to a very important issue. The argument goes as follows. A computer is a device of a dual nature, both as a machine and as a medium. Its basic characteristic is that it can be programmed. This means that lots of different categories of computer users would benefit from attaining some programming skills. Unfortunately, lots of students find programming courses too abstract and difficult. This leads to a conclusion that some other means of communicating programming skills than classical programming courses for computer science students have to be developed. Thus, in the work of Andersen et al., narratives are a part of a game-based multimedia environment used to teach multimedia students programming skills needed to program similar multimedia environments. This means that they are not used for increasing motivation of students only, but more like ToonTalk or Magical Forest for small children – to present some new knowledge in a form that the students would find naturally interesting.

I would argue that this argument goes beyond multimedia students only. In lot of different professional activities people could benefit from knowing some basic programming skills – laboratory technicians, designers, administrative stuff, just to name randomly some different professions. This argument could become only stronger in the future, taking into account development of ubiquitous computing and omnipresence of microcontroller-based devices everywhere around us. I would even argue that, in order to cope with future world, everyone should get some programming skills, somewhere at high school level, at least at the level of manipulating spreadsheets and adjusting rules and properties of abstract objects in some environment.

To check the assumption that narratives could play a beneficial role in programming education, a small case-study over two years was conducted. Two different teams, consisting of 7 children each, ages 11-15 years, were coached and observed while constructing and programming LEGO robots for FIRST Lego League competition. During the first year narratives were used only as motivating factor and sense-making of a programming task. During second year, roles of variables in a narrative way were used to make children understand some basic algorithms.

This paper is organized as follows. In the next two sections concept of roles of variables and a short overview of RoboLab environment are presented. These two sections are included for readers unfamiliar with one or both of these concepts; other readers could skip them. More elaborate presentation of roles of variables [17-21] and RoboLab [6,9] could be found in the cited sources. In the Section 4 main results of the conducted case-study are presented. Using narratives for making sense of a programming task is described, as well as using roles of variables in a narrative way in order to make children understand some basic algorithms. The last section is a summary of the work.

Previous Work: Roles of Variables

In order to achieve the goal described in the Introduction, some new educational tools that would appeal to a broad range of students, and not only those technology inclined, need to be developed. I believe that narratives could play an important role in these tools. A search of existing programming curricula lead me to a very interesting concept of Roles of variables, under development from 2002, at the Department of Computer Science, University of Joensuu, Finland. Because of their importance for narrative learning environments, a short presentation of this concept will be outlined here.

Roles of variables are a novel approach in teaching programming, started about 2002 in Finland – see, for example, [17,18,18,19], and web-site of the project [21]. The argument for their introduction (adapted from the web-site), goes like this. Knowledge about computer programming covers the following three categories:

Table 1. Categories of computer programming knowledge [21]

programming language knowledge	the syntax and semantics of some certain language (e.g., how an assignment statement is written and what effect it has)
program knowledge	knowledge about a specific program
programming knowledge	how to construct programs from abstract concepts within the programming paradigm in use (e.g., variables, iteration etc. in procedural programming)

It can be easily agreed that the programming knowledge is the most important, as it is independent on the specific language. It is difficult for many of the students to construct and understand the programming knowledge, as programming is typically taught by analyzing existing programs and structures of a specific language.

Roles of variables are programming knowledge that can be explicitly taught to students. As authors point in [17], from their throughout analysis of programs used to teach novices programming, 99% of variables belong to one of the categories presented in Table 2.

This representation of variables is narrative in its core meaning, almost making a theatre-performance out of a computer program. Instead of abstract value-holders, we now have players that are interacting with each other and influencing each other, while being obliged to follow certain rules and restrictions, like in lots of children games scenarios (“You cannot move now”, “Now it’s your turn to run!”).

Table 2. Definitions for roles of variables [21]

Variable role	Definition
Fixed value	A variable whose value is not changed in run-time after initialization
Stepper	A variable going through a succession of values in some systematic way, depending on its own previous value and possibly on other steppers, stepper followers and fixed values.
Most-recent holder	A variable is a most-recent holder, if its value is the latest gone through value of a certain group or simply the latest input value.
Most-wanted holder	The value of a most-wanted holder is the "best" or otherwise the most-wanted value out of the values gone through so far. There exist no restrictions for measuring the superiority of the values: the most-wanted can mean, for example, the smallest or the biggest number or a number closest to a certain value.
Gatherer	The value of a gatherer accumulates all the values gone through so far.
Transformation	A transformation is a variable that gets its new value always with the same calculation from value(s) of other variable(s).
One-way flag	A one-way flag is a Boolean variable which once changed cannot get its original value anymore
Follower	Always gets the old value of another known variable as its new value
Temporary	A variable is a temporary if its value is always needed only for a very short period.
Organizer	An organizer is an array which is used for reorganizing its elements after initialization.

Authors used this approach to teach text-based programming languages, like C, Java and Pascal to university undergraduate students.

This inspired me to try their approach with 11-15 years old teenagers and with an iconic programming language specially aimed for children, namely already mentioned RoboLab for programming LEGO Mindstorms robots [6]. My initial results were encouraging, thus allowing for yet another confirmation of an assumption that "storyfication" could help some users to make sense of abstract technological environments [3, 4, 7, 8, 10].

RoboLab A-B-C

RoboLab is a graphical programming language for LEGO Mindstorms, developed at Tufts University, and aimed at school environments. It is based on National Instruments professional LabView software (<http://www.ni.com/labview/>). RoboLab is not a free software. It needs to be purchased via special school suppliers (http://www.lego.com/education/default.asp?page=5&l2id=8_1). Both individual and school licences are available. For more information and documentation about RoboLab please see <http://130.64.87.22/robolabatceeo/> and [9].

Here will be described just very basic of RoboLab programming, in order to enable the reader, who never met RoboLab before, to follow this presentation independently.

A RoboLab program is a set of icons, picked up from a menu, and connected with wires. Icons are abstractions for actions to be performed on robot's motors, or decisions made after readings from different sensors, and wires represent program flow.

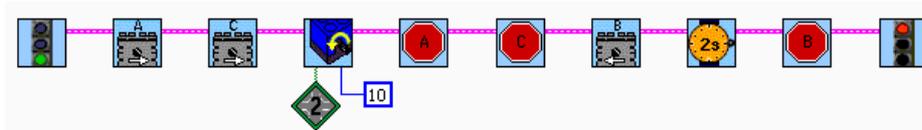
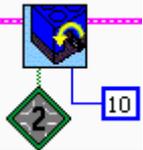


Figure 1. A simple RoboLab program

The program on Figure 1 means:

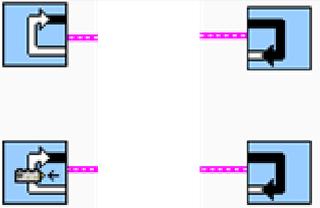
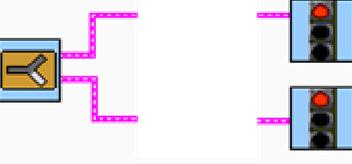
	Start the program
	Start motor A
	Start Motor C
	Wait until rotation sensor 2 shows value 10 (motors A and C are running in the meanwhile)
	Stop Motor A
	Stop motor C
	Start motor B

	Wait for 2 seconds (motor B is running in the meanwhile)
	Stop motor B
	Stop the program

This is enough to complete some simple robot tasks – typically motors A and C are connected with wheels of the robot and motor B with some tool – so this program could mean that robot goes straight for a prescribed distance, and then does some task by moving the tool for 2 seconds. This is enough to solve several tasks on FLL competitions, and this is what majority of teams typically use.

However, RoboLab is not a “toy” language. It is pretty complete programming language, which allows for majority of program control structures available in other programming languages, presented in Table 3.

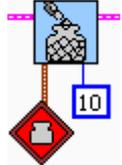
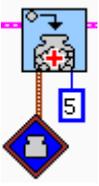
Table 3. Icons for control structures in RoboLab

	<p>If-Then structure. If condition is graphically represented – here it is a test whether a touch sensor is pressed or not. By nesting these structures, it is possible to represent If-Then-Else structure.</p>
	<p>Do – structure. All commands between the bounding two are repeated for certain amount of iterations, or while certain condition is (is not) fulfilled</p>
	<p>Multitasking icon – several tasks could be run in parallel, and each of them needs its own stop.</p>
	<p>Event (interrupt) monitoring and jump to a place for event code execution</p>

One of main and on-going challenges since RoboLab’s creation was how to explain these control structures to children – as they are basically the same as in other “serious” languages taught to students at high school / university courses.

Another issue in RoboLab is how to represent variables. The chosen representation is the one of a container – and an icon suggests a jar filled with something. The content of the jar can be preserved, and taken out when needed. This representation is excellent for the beginning – for saving data from sensor(s), and using data at some later point. This representation also allows for basic arithmetic operations, as could be seen from Table 4. However, it is very cumbersome for any more involving calculations.

Table 4. Icons for variables and arithmetic operations in RoboLab

	<p>Basic container functionality. Containers are differentiated by colours. Icons represented here mean that constant “10” is saved in red container.</p>
	<p>Adding 5 to whatever is already in blue container.</p>
	<p>Adding value in Red container to value in blue container, and keeping the result in blue container.</p>

Case study: FIRST LEGO League competition

FIRST LEGO League (FLL) is an international program for children aged 9-14 (9-16 in Europe) that combines a hands-on, interactive robotics program with a sports-like atmosphere, see web-site <http://www.firstlegoleague.org>. Teams consist of up to 10 players with an adult coach. Each September, a new Challenge is unveiled to FLL International teams across the world. Over the course of 8 weeks, teams strategize,

design, build, program, test and refine a fully autonomous robot capable of completing the various missions of that years challenge, using LEGO Mindstorms programmable sets.

FLL competition is an excellent test-bed for studying how children collaboratively approach challenging technological problems. For several years I served as a technical judge for FLL tournaments, judging teams building and programming achievements. Although it was exciting to watch children's constructions in action, I was somehow disappointed with the programs children wrote – majority of the teams did not use anything significantly more complex than the program on Figure 1. I wanted to have hands-on explanation for that situation, and I decided to serve as a team-coach and observe children building and programming their robots. During 2004/2005 school year, I coached a team consisting of 7 girls, 11-15 years of age, without previous robot building and programming skills, preparing for FLL tournament. During 2005/2006 school year, I co-coached another team of 7 children, both boys and girls, 11-14 years old. Some of the children from the second team have some difficulties at school, and attend some special classes.

Narrative elements in making sense and giving motivation in general FLL challenge

Challenges for FLL tournaments are different every year, and they are chosen with a constructivist approach to learning and problem solving in mind. This means that challenges do not have unique solutions, and that each team should choose its own strategy. Competition itself is organized as a sport event, with lots of audience cheering for robots (!).

But, from the other side, it is quite possible to look at FLL challenges from a narrative perspective. The robot, which is a main character, has to go into the world, to overcome some obstacles and to solve certain tasks. Robots tasks typically involve placing some objects into the world, and obtaining some other objects from the world to the specific place on the playing board called the base (this could be the fairytale castle). The robot could even get some helpers, in form of sensors, that tell him some information, and tools, that perform certain actions. If we compare this scenario to a typical fairytale plot, for example from Propp, we see an almost 1:1 match. This is even truer if we take into account evidences from literature, [1] and [5] for example, that children tend to look at robots as if they are their “almost alive” pets and friends. This is also validated by my observations with the FLL team, where the girls named their robot “Rainbow Baby”, because it was very colourful, and tools to solve certain missions “Ball-pal” and “The little fellow”. The teams are asked to write a technical documentation about how they constructed and programmed their robots to solve the missions, and when the girls needed to describe why they changed from caterpillars to wheels, they have written: “Rainbow Baby is now much faster. But it has also grown a little taller, our baby is growing up, and we are so proud!” [12].

All these just add to evidence that if faced with a technical challenge placed in a convenient story which makes sense for them; children are willing to put lots of efforts into solving it (and learning lots of the things on the way).

Narratives as making sense of programming for non-tech children

As a technical judge on several FLL competitions, I have noticed that programs that majority of teams write for the competitions are very simple. Lots of teams use just “Motor start – motor stop after a certain amount of time” commands, some teams use “run motors until something happens with a sensor” – for example until a touch sensor is pressed, a light sensor shows value greater than a threshold, or a rotational sensor shows certain amount of rotations. Very few teams use more complex programming structures, like cases, loops, and variables, although RoboLab environment offers all of them, and much more.

Observing several teams working with the challenges, it becomes clear that children do not feel a need to learn more programming – they want to build a robot, tell it what to do, and are too busy for learning abstract concepts that they do not see immediate use for. Currently, although “Mindstorms Ultimate Builder Set” offers great inspiration for robot building, there are no other multimedia tools which would introduce children into more powerful RoboLab programming but RoboLab itself.

Thus, in order to motivate the all-girls team to learn more programming, I decided to amaze the girls with something, in order to show them power of variables in programs. This is from their Technical diary [12]:

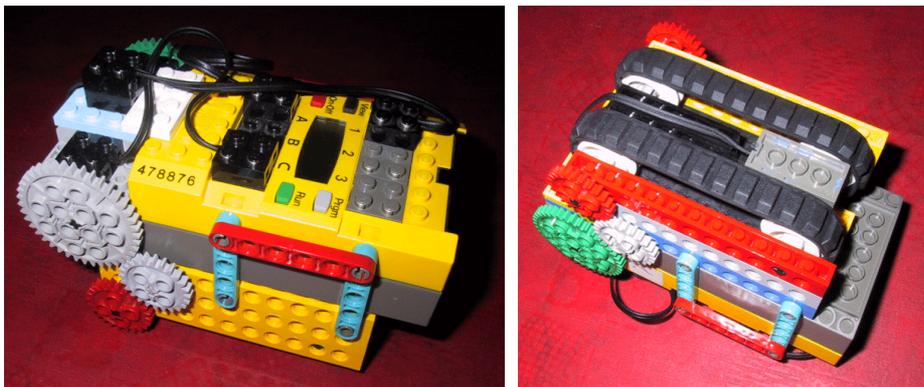


Figure 2. “This is the robot we used for our lection in variables” [12]

...”Our coach took the small robot we built some time ago, while playing with small caterpillars. Its wheels are so close together that the possibility of turning

equals zero! But here you can see it has the rotational sensor. The little robot, as you can see, has no other sensors but that rotational one.

Anyhow, when it runs straight and bumps into a wall, it beeps and goes backward, until it hits something else, this time from the back side. Than it beeps again and goes to the front until it bumps into something, and so on. We tried to place obstacles on different distances, but nothing could confuse our little friend – when it bumps into something, independently from which side, it beeps and changes the direction of its motion. As we knew from before, you would probably need two touch sensors to program this – but the little robot has only one sensor, and it is rotational. Even more, if we hold the robot by hand, or press it with something from the top, it also beeps and tries to move into the opposite direction. Amazing!

Our coach showed us the Containers Menu in RoboLab, made us read the Help, and challenged us to come up with a program that can reproduce the behavior we had just seen. It was not easy, and she needed to help us some more. After a while we succeeded to make it. You can see it printed here.”...

The program the girls made is on Figure 3. The upper line has a loop that will run forever, and in the loop commands to run the motors forward and backward. The direction of motors is changed when the subroutine, represented with an icon with a green circle, is finished with its execution. Subroutine basically compares two consecutive readings from the rotational sensor, taken within half a second interval, as long as this difference is different from 0. If rotational sensor shows same values in different moments, that mean that the sensor is not rotating, i.e. that robot got stopped – probably because it bumped into something. Before writing this program, the girls were familiar with loops, forks (cases) and subroutines, but variables (containers) were completely new for them, and this task was difficult. Only 3 out of 7 girls picked up this programming challenge and showed some understanding. Those 3 girls are good mathematicians and are interested in computers and technology anyhow, but this was very difficult concept for them to grasp. At this moment I understood that programming of LEGO robots, without proper programming training and just with “learning by doing” approach, might be a very challenging task for majority of teams. That is also a partial explanation for simplicity of majority of children’s programs.

At the start of FLL 2005/2006 season, I wanted to inspire another team, consisting of 4 boys and 3 girls, aged 11-14, to learn some more programming. I wanted to use the same robot and the same program, but before showing it to children, I did the analysis of the program from Roles of Variables perspective, presented in previous chapter. My observations are in bobbles below program print, on Figure 3.

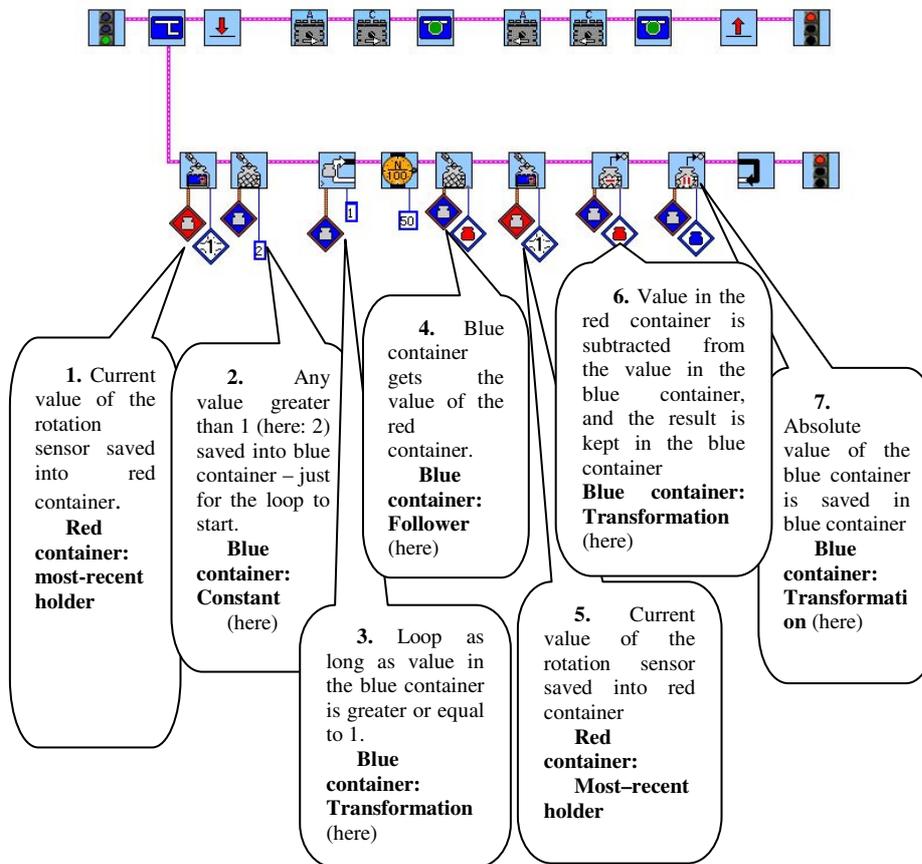


Figure 3. RoboLab program which uses containers to determine if the robot is running or it has bumped into something

Analyzing this program from Roles of Variables point of view, some sources of students’ confusion reveal themselves: while the Red container is clearly a Most-recent holder, the blue container first gets some mysterious constant value, and then starts to change roles between a Follower and a Transformation.

I wanted to test how would children understand the program if every variable should have a clear single role. For this program, the following roles are needed:

- Most-recent holder, to keep the value of the latest reading from rotational sensor;
- Follower, to keep the value of the rotational sensor from the previous moment;
- Transformation – a variable which should get new value every half a second, equal to absolute difference of the Most-recent holder and its follower. The

same variable should be used to terminate the loop – if the follower is equal to the most-recent holder, the robot is not moving.

The issue in realizing this idea was that the concept of variables as containers and calculations with them in RoboLab does not support phrases like

$a := b + c.$

Only

$a := a + b$

is supported.

While paradigm $a := a + b$ is perfectly correct one, especially in the field of embedded systems programming, where resources are scarce, I doubt that it is the best one for introductory computer courses, especially for young teenagers, whose abstract thinking skills just started to develop. There is also a vast literature supporting this, for an overview please see [22].

Fortunately, RoboLab allows a concept of a “Sub-Vi”, which could be thought of as a macro in textual programming languages. In other words, it allows that a group of icons is represented with a single icon, and users have a freedom to design an icon for the new Sub-vi. Thus, in order to calculate a value of the Transformation (in Yellow container) by subtracting the Most-recent holder (Red container) from its Follower (Blue container), I had to introduce an icon on the right hand-side of the Figure 4, instead of two icons on the left-hand side of the same Figure. I made this in order not to confuse the learners with a concept of $a:=b$ (i.e. yellow container gets the value of the blue container), as this is just a specificity of an environment, and represents just an obstacle to understand more important concepts.

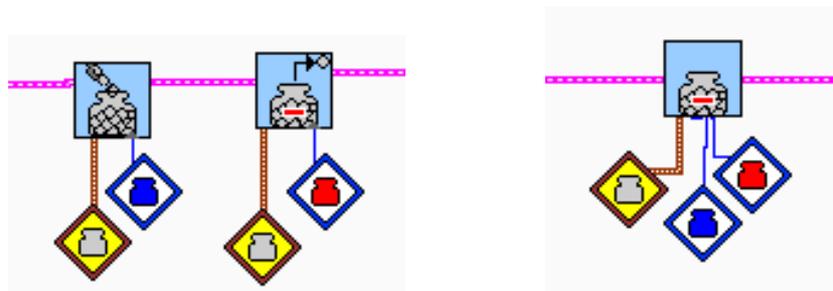


Figure 4. A way to write expression $a=b-c$ in RoboLab: $a:=b$ and $a:=a-c$ (left), and a single icon which hides these two operations from the user (right)

After these adjustments, the re-written program with identical functionality is presented on Figure 5.

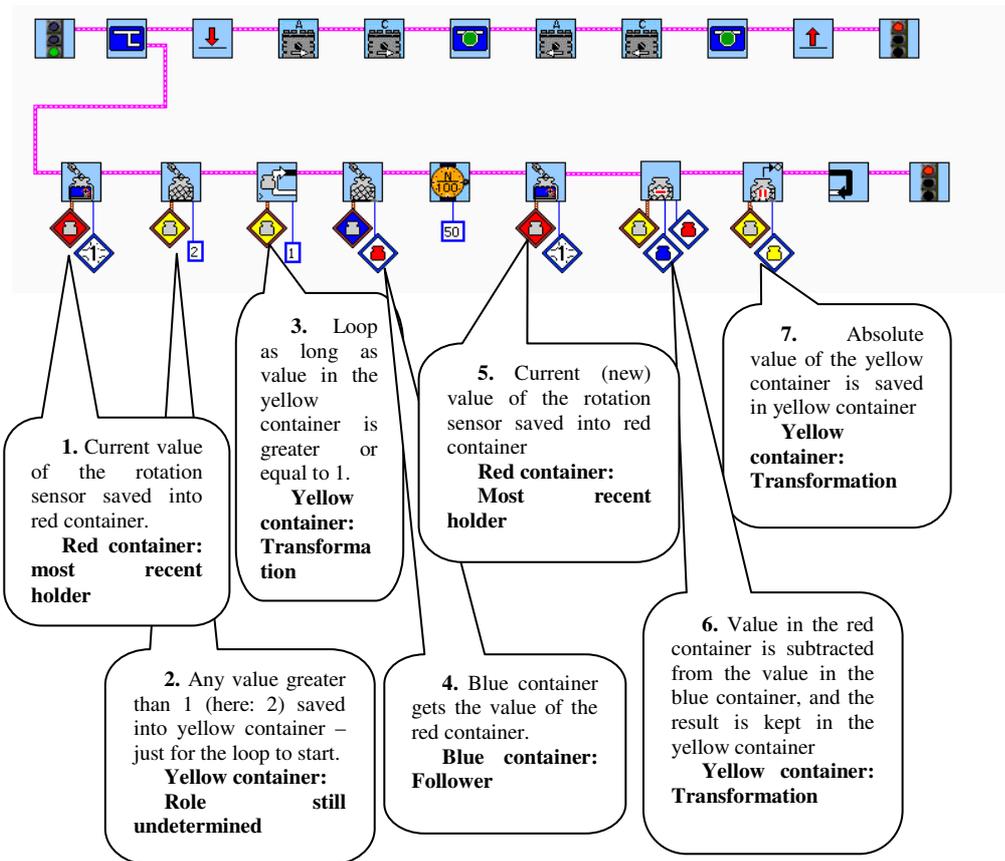


Figure 5. RoboLab program where each variable has a clear role

I wanted to present this program to a group of 7 children (11-14 years old, 4 boys and 3 girls), preparing for FLL competition. Children were not very interested in programming and learning, and half of them had problems to follow normal lessons at school, so they were getting professional help. Thus an idea to ask them to develop a program themselves, like previous year, would not work. We discussed the program, and as children had difficulties to understand, I assigned roles to 3 of the children:

- to remember the value that rotation sensor is reading NOW (“most recent value” - red container);
- to remember the value that the rotation sensor was reading half a second ago (“follower” - blue container)

- to calculate and remember difference between blue and red containers (“transformation” - yellow container).

Then we played role-game, like in theatre. I would move robot for a distance, and say “Now” – and all of the actors should tell me their current values – by reading new value from robot’s display and performing assigned roles. This way, after a while, I got an “Aha!” effect from 6 of them. All of a sudden, instead of being just an abstract concept of a value-holder, that changes values who-knows-how and who-knows-when, Red, Blue and Yellow container become like small magic helpers of a hero (i.e. the robot), with their precisely defined roles, that tell the robot what happens around. This is like a fairy-tale scenario, familiar to children.

General conclusions cannot be derived from this isolated experiment, but it indicates that using storytelling techniques and roles of variables concept, more average and below-average children could become more familiar with programming concepts.

It is important to note that in this scenario, storytelling is not “sugaring the pill”, but represents a crucial role in reaching understanding for children who would otherwise be left behind.

Conclusion

The aim of future work in this area is to investigate and figure out how to develop relevant “technical” stories and how to integrate them best in multimedia high-tech environments of edutainment LEGO toys, especially to investigate how far “Roles of Variables” concept could be pushed in RoboLab-like programming environments (including new LEGO NXT programming language), and figure out with more scientific approach could this approach help more different children to understand some basic of programming.

I do not believe that “narrationalizing” (or “storifying”) the processes of building and programming LEGO models is impossible or a not desirable process. I do believe that narratives could play an important role in the LEGO technology universe, by:

- giving and keeping motivation;
- providing initial and “just in time” instructions as necessary (teaching users essential skills required to start and continue the process);
- making the instructive experience of playing with technological toys more accessible and enjoyable to more kids, not only to those “technology-oriented freaks”.

Results of the presented case-studies strongly support this belief.

References

1. B. Bartlett, V. Estivill-Castro, S. Seymon, Dogs or Robots: Why Children see them as Robotic Pets rather than Canine Machines?, 5th Australian User-Interface Conference, AUIC04, Dunedin, Australia, 2004
2. P. Bogh Andersen, J. Bennedsen, S. Brandorff, M. E. Caspersen and J. Mosegaard, Teaching Programming to Liberal Arts Students — A Narrative Media Approach. *ITiCSEV3*, , Thessaloniki, Greece, 2003
3. A. Doxiadis, Embedding mathematics in the soul: narrative as a force in mathematics education. *Opening address to the Third Mediterranean Conference of Mathematics Education*, Athens, January 3, 2003
4. A. Doxiadis, The Mystery of the Black Knight's Noetherian Ring - An investigation into the story-mathematics connection with a small detour through chess country., *A keynote address to the Fields Symposium on Online mathematical investigation as a narrative experience*, Faculty of Education, University of Western Ontario, 11-13 June 2004
5. A. Druin and J. Hendler, editors, *Robots for Kids – Exploring new Technologies for Learning*, Morgan Kaufmann Publishers, 2004
6. LEGO Company, Using RoboLab, LEGO Mindstorms for schools, 2004
7. L. M. Olivieri, High school environments and girls' interest in computer science, *ACM SIGCSE Bulletin*, Volume 37 , Issue 2, Pages: 85 – 88, 2005
8. C. Rader, C. Brand and C. Lewis, Degrees of Comprehension: Children's Understanding of a Visual Programming Environment. *CHI conference*, Atlanta, GA, 1997
9. Robolab, <http://www.lego.com/eng/education/mindstorms> (visited 16th November, 2005).
10. Y. Solomon, J. O'Neill, Mathematics and Narrative. *Language And Education* Vol. 12, No. 3, 1998
11. Waraich, Using Narrative as a Motivating device to Teach Binary Arithmetic and Logic Gates. *ITiCSE 04*, Leeds, UK, 2004
12. WoW FLL Technical diary, unpublished material, 2005.
13. ToonTalk, <http://www.toontalk.com/> (visited 20th September 2005)
14. Magical Forest (Floresta Magica, in Portuguese), <http://educacao.cnotinfor.pt/index.php?lng=pt&pag=123&id=0>, (visited 20th September 2005)
15. Playground project, <http://www.ioe.ac.uk/playground>, (visited 20th September 2005)
16. Imagine LOGO, <http://www.logo.com/imagine/>, (visited 20th September 2005)
17. J. Sajaniemi, M. Kuittinen, Program Animation Based on the Roles of Variables. *Proceedings ACM 2003 Symposium on Software Visualization (SoftVis 2003)*, San Diego, CA, June 2003. Association for Computing Machinery, 7-16. Best Paper Award.
18. M. Ben-Ari and J. Sajaniemi, Roles of Variables as Seen by CS Educators. *ITiCSE 2004*, Proceedings of the 9th Annual Conference on Innovation and Technology in Computer Science Education, Leeds, UK, June 2004.
19. M. Kuittinen and J. Sajaniemi. Teaching Roles of Variables in Elementary Programming Courses. *ITiCSE 2004*, Proceedings of the 9th Annual Conference on Innovation and Technology in Computer Science Education, Leeds, UK, June 2004.

20. T. Stützle and J. Sajaniemi. An Empirical Evaluation of Visual Metaphors in the Animation of Roles of Variables. *Informing Science Journal*, 8, 87-100. (Also presented at the 2005 Informing Science and Information Technology Education Joint Conference (InSITE 2005), Flagstaff, AZ, June 2005, as a Best Paper.)
21. Roles of Variables web-site: http://www.cs.joensuu.fi/~saja/var_roles/ (visited 20th September 2005)
22. Kelleher and R. Pausch, Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers, *ACM Computing Surveys*, Vol. 37, No. 2, June 2005, pp. 83–137.