Learning Algorithmic Thinking with Tangible Objects Eases Transition to Computer Programming

Gerald Futschek, Julia Moschitz

Vienna University of Technology, Institute of Software Engineering & Interactive Systems,
Karlsplatz 13, 1040 Vienna, Austria
{Gerald.Futschek, Julia.Moschitz}@ifs.tuwien.ac.at

Abstract. Learning algorithmic thinking can start in early years and must be oriented on the thinking ability of young children. Suitable environments with tangible objects and easy to understand problems motivate the young to learn the first concepts of algorithms. We present in this paper a learning scenario *Tim the Train* for primary school children, that involves tangible objects and allows a variety of interesting tasks to learn basic concepts of algorithmic thinking. We also show how a smooth transition from a playful environment with tangible objects to a virtual Scratch/BYOB environment may help the young learners to learn their first steps in understanding virtual environments and programming concepts.

Keywords: Algorithmic thinking, primary education, explorative learning, tangible objects, learning by doing

Fig. 1. *Tim the Train* with tangible parts made of coloured wood

1 Introduction

At the beginning of learning algorithmic thinking learners often have problems with this abstract type of thinking and with the abstraction in a programming environment. Therefore, we try to build a bridge between the real world and a programming environment. *Tim the Train* is an example which helps especially beginners to pass the river between the tangible world and the abstract world of programming environments and languages.

Many efforts to create environments and learning scenarios have been made in order to make learning algorithmic thinking and programming easier, more pleasant and more efficient. Some more prominent programming environments that are suitable also for very young children are Logo, Alice, Baltie and Scratch [1,2,3,4]. All these environments allow the manipulation of virtual objects with a wide variety of commands. Learning environments should try to attract as many learners as possible by appealing to different senses and learning styles. For young learners the kinesthetic aspect of learning is very important, because they learn by touching, seeing, hearing, feeling and smelling [5]. Kinaesthetic elements are very important even for engineering education in higher education, because the students are invited to be active with more than one sense in the learning process [6]. Some programming environments involve tangible objects, the mechanical Logo turtle and programmable robots are well-known examples [7].

In this paper we propose a learning scenario with real tangible objects that is suitable to learn the basic concepts of algorithmic thinking. It has a limited number of commands that are easy to understand. A simulation of this scenario in Scratch allows a smooth transition to virtual learning environments and the world of computer programming.

2 Learning Scenario *Tim the Train*

Tim the Train describes a bin packing scenario. The containers of a train should be loaded with parts of different shapes. No container should be overloaded. This learning scenario provides the opportunity to come in contact with a more advanced problem of computer science. *Tim the Train* helps beginners to train algorithmic thinking in a playful way and to understand some aspects of the bin packing problem as well.

The implementation of *Tim the Train* consists of a set of wooden elements for train, containers and load parts (see Fig. 1) and also of a simulation in Scratch/BYOB. *Tim the Train* includes a locomotive, containers, parts in different shapes and sizes and a set of symbolic commands. The colours of the train are chosen so that they are attractive for boys and girls. The wooden train and the simulation in Scratch/BYOB use the same structure and colours, so that a recognition factor is given, which helps beginners to get a feeling for the simulation environment on the computer.

A variety of interesting tasks can be performed with this learning scenario. Basically, learners do not need special pre-knowledge to solve these assignments. The game-like tasks are suitable for children from the age of 5 years onwards. Reading is

not necessarily required, because the instructions for the algorithms are marked with symbols and words. The teacher should explain the tasks and the meaning of the commands anyway. The number of symbols for describing the commands (a sort of programming language) is kept small to help the learners to focus on the basic algorithmic concepts. The learners get a card with explanations about the actions and their acceptation. The programming language is easily expandable, because the language is realized on cards. New command cards may be created by the learners themselves. For advanced learners it is a challenge to create new actions or forms of loops and suitable symbols for these actions and loops.

The problems (tasks) are chosen in a way that they are interesting for beginners as well as for advanced learners, because the tasks provide different levels of difficulties. Advanced learners can solve the easier problems too in order to revise and consolidate the basic skills of algorithmic thinking. However, the material is also appropriate to adolescents and adults who want to learn algorithmic thinking in a playful way.

3 Basic Algorithmic Concepts

Learning concepts is much more important than learning systems or programming languages. At the very beginning of learning algorithmic thinking we can observe very basic concepts that should be involved in learning scenarios:

Basic commands – basic actions
Sequence of commands
Alternative of commands (if)
Iteration of commands (loop)
Abstraction command (method)

Commands are usually represented by symbols or by text. The main purpose of a command is that when executed it invokes well defined actions. Each basic command is related to a basic action in the learning scenario. So a learner has to understand that a command is an abstraction of an action. The concepts sequence, alternative and iteration define the order in which commands are executed. These concepts are essential in algorithmic thinking.

Other important algorithmic concepts like recursion, parameters, variables, data types are omitted intentionally when addressing very young learners. These concepts need deeper abstract thinking skills and are therefore suitable only for elder learners or more advanced learners.

4 Interesting Tasks

This chapter describes three different tasks for beginners to play/learn with *Tim the Train* that lead to first algorithmic thinking skills. The level of difficulty increases from task to task. These three tasks are just examples of tasks that can be given. A teacher may vary these tasks or can pose even his own tasks.

4.1 Playing a given algorithm

In the first level the beginners have to reproduce a given algorithm. By this way the beginners familiarize themselves with the symbols (commands) of *Tim the Train*. Additionally, the learners try to play (execute) the given algorithm and to find out the result of the given algorithm. This game is appropriate for beginners who have no or only few experiences of algorithmic thinking. This problem is also adequate for very young children, who cannot read or write. At the beginning the containers of the train (Fig. 2) contain some parts to show what the action "reset" does at the beginning.

Fig. 2. A train at the beginning of the algorithm

The following table shows the parts and the order of their appearance.

1st part	2nd Part	3rd part	4th part	5th part	6th part
					

Table 1. A sequence of incoming parts

The first command (reset) of the algorithm (Fig. 3) is to clear all containers of the train. Then the train is empty and the first part can be dropped. The instruction "drop" means to drop the part at the leftmost position where the part would reach the deepest place. This step will be repeated four times. After these drops the last part overloads the wagon therefore it is lifted again. Then the next wagon is loaded with three parts.

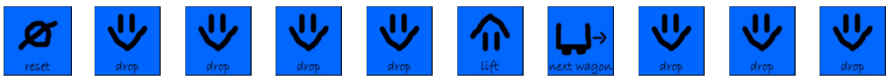

Fig. 3. Example of a given algorithm

The learners have two possibilities to build the solution. On the one hand they can play the algorithm with the wooden train, on the other hand they can even use the simulation in Scratch/BYOB to find the solution. When the algorithm is finished, one can find the following final state (Fig. 4):

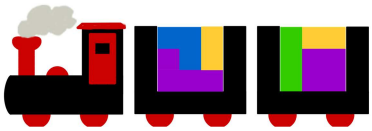

Fig. 4. The train at the end of the algorithm

4.2 Second Task: Writing an algorithm

In the second task, the result (Fig. 5) is given and learners have to find out the algorithm that leads to the given result.

Fig. 5. The final state to be achieved

For beginners a set of instructions cards (Fig. 6), which are required in the algorithm, may be given.

Fig. 6. A given set of instruction cards

As additional help to find out the basics of the algorithm, a few tips may help: “At first you have to find out in which order the parts are fitted in the wagons.”, “Check the parts, if they are in the right position. Maybe you have to flip or rotate some of the parts.”

Advanced learners may begin to solve the problem without the tips and the set of given commands. As an additional task, advanced learners may try to find a better solution by use of iterations.

If the learners use the actions shown in Fig. 6, the parts have to come in the following order and orientation:

1st part	2nd part	3rd part	4th part	5th part	6th part
					

Table 2. The sequence of incoming parts

The solution with the instructions given is without any use of loops. Therefore, it will be easy to solve for beginners (Fig. 7). The beginners can think about: What is happening when the parts are in another order or orientation? How must the algorithm

be changed to reach the given result? Does the algorithm always work for all possible input sequence of parts? These questions are instructive for beginners, because they will recognize by this way that the solution is only effective for the given sequence of parts and not a universally valid solution for this kind of problem.

Fig. 7. A possible solution for the required algorithm

4.3 First algorithm with loops

The next task allows beginners to develop repeat–loops in an easy way. The didactic approach is that the learners develop the loop step by step. At first the learners have to find out what a given algorithm does and then they have to reformulate the given algorithm using the loop construct. Next they have to find out: "Under what condition is the algorithm correct and under what condition not?" This question should help especially beginners to think about the differences between an algorithm which is generally effective and one that is effective only for a specific problem. The precondition for the algorithm of Fig. 8 is that the train has at least three wagons and overloading is possible. As additional difficulty, the learners do not know in which order the parts are coming. The following algorithm drops a part in each of the first three wagons. This action is repeated three times.

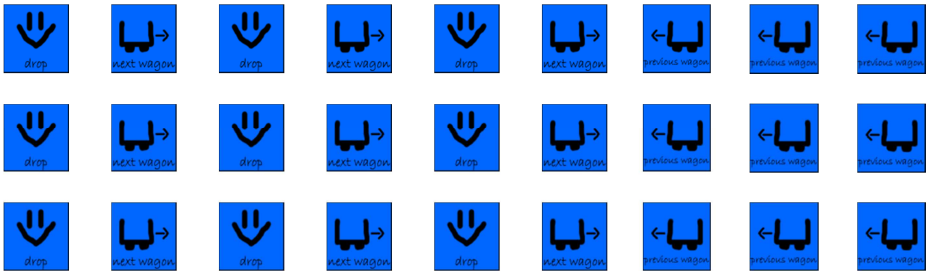

Fig. 8. A given simple algorithm at the beginning of the task

At first the learners have to find out which parts of the algorithm are repeated. It is easy to see that the sequence of commands contained in the 3 rows is exactly the same. But also inside each row one can find repetitions. The left block in Fig. 9 shows the actions "drop and go to next wagon" which are repeated 3 times. The right block in Fig. 9 shows that the action "go to the previous wagon" is also repeated three times.

Fig. 9. The repeated parts of each row are marked by coloured blocks

As intermediate step the learners may build an algorithm using loops only for the two blocks in Fig.9, because loops, especially nested loops, are not so easy to understand. For beginners the breaking points of loop understanding are to find the repeated actions and to set the right limitation of the loop.

An example of a solution is given in the following picture (Fig. 10). This solution shows explicitly, that the two loops of the first line are repeated three times.

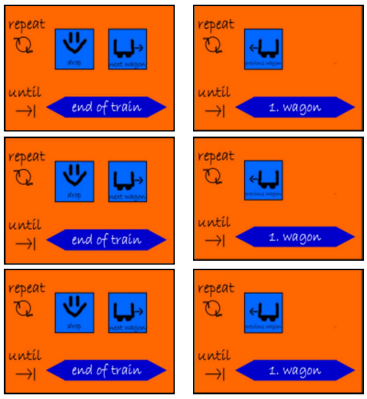

Fig. 10. An algorithm with six loops

Additionally, by this way the learners discover that they can simplify the solution once again and develop their first nested loop in an easy way. A solution with an additional loop may simplify this algorithm even further, see (Fig. 11).

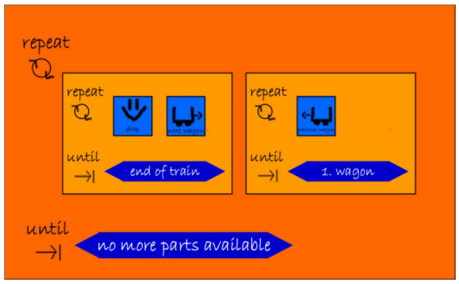

Fig. 11. A solution with nested loops

This example shows how loops can be introduced to beginners. Step by step the learners develop new versions of algorithms.

As additional tasks it would be interesting to write algorithms to fill a wagon and to empty a wagon. These loops are easy to build, but they are good exercises to develop skills in loop understanding. The algorithm to empty a wagon is simple, because one needs only the action “lift” and the condition “empty” (Fig. 12).

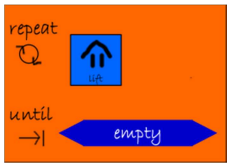

Fig. 12. Emptying a wagon

Filling a wagon is only slightly more difficult. The additional difficulty is to detect that a wagon cannot be filled any more. A possible solution is to drop parts until the wagon is overloaded. Then the last dropped part has to be lifted back (Fig. 13).

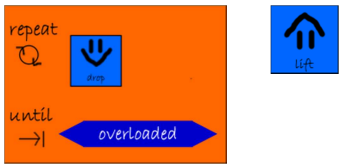

Fig. 13. Filling a wagon

5 Learning experiences

All the examples described in the last chapter try to help beginners to train their competences in algorithmic thinking. Although the complexity of *Tim the Train* is simple, it has a lot of possibilities to construct interesting tasks for different levels of algorithmic thinking skills.

Fig. 14. Filling the wagons with tangible parts

For learners who have problems with algorithmic thinking or who are using preferably visual or kinaesthetic learning styles, the wooden train helps them to understand the problem and eases finding a solution (see Fig. 14). This type of learners is grateful, when they can touch and flip the real parts in the real world or to execute an algorithm without a computer environment. At the beginning these

learners are often overstrained with doing both: algorithmic thinking and at the same time orienting in a new abstract programming environment.

We strongly recommend that the learners work in groups. With help of the wooden train the learners can work together, can discuss their problems, and they can find solutions to their problems faster.

In the examples described *Tim the Train* is standing and a crane is moving from one container to another. Advanced and more talented learners like to find new tasks or to expand the set of commands. For example, it would be interesting to move the crane or the train in both directions. This learning scenario would enable a lot of further solutions and a basis for discussions of the different solutions.

Advanced learners are highly motivated to implement their commands also in Scratch or BYOB [8]. The principles of algorithmic thinking are in a virtual programming environment like Scratch or BYOB the same as with tangible objects. It differs in the granularity of the basic actions. Even the basic commands of the wooden model have to be programmed, see Fig. 15.

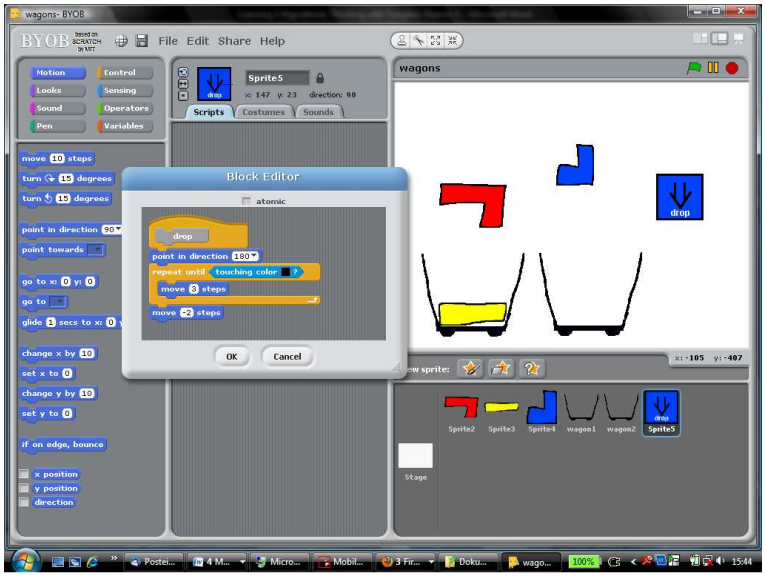

Fig. 15. Implementation of command “drop” in BYOB

The advantage of this learning scenario is that the learners may decide between different learning environments adapted to their individual learning styles and types.

6 Conclusions

Tim the Train is a physical micro world which helps learners to train algorithmic thinking and helps them to have a smooth transition to virtual learning environments and the world of computer programming. The children learn about the basics of the

bin packing problem and make their first steps in the programming environment of Scratch.

With *Tim the Train* the beginners are motivated to do further programming in different environments. This holds even for those learners who are not absolutely ecstatic of programming at the beginning.

References

1. Papert, S.: Mindstorms: Children, computers, and powerful ideas (1980)
2. Cooper, St., Dann, W., Pausch, R.: Teaching Objects-first. In Introductory Computer Science, Proceedings of the 34th SIGCSE technical symposium on Computer science education, pp.191-195 (2003)
3. Baltie home page, http://progopedia.com/language/baltie/
4. Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., and Resnick, M: Scratch: A Sneak Preview. Second International Conference on Creating, Connecting, and Collaborating through Computing. Kyoto, Japan, pp. 104-109 (2004)
5. Hayes, D.: Encyclopedia of Primary School. Routledge Chapman & Hall, pp.231-232 (2010)
6. Felder, R., Silverman: Learning and Teaching Styles in Engineering Education. In: Journal of Engineering Education, vol 78(7), pp.674 -681 (1988)
7. Horn, M., Jacob, R.,: Designing Tangible Programming Languages for Classroom Use. In: 1st international conference on tangible and embedded interaction, pp. 159-162, ACM, New York (2007)
8. BYOB - Build Your Own Blocks, An extension of Scratch to program own commands: http://byob.berkeley.edu/