Visual Programming Language for Thymio II Robot

Jiwon Shin*
Dept. of Computer Science
ETH Zürich, Switzerland
jiwon.shin@inf.ethz.ch

Roland Siegwart
Autonomous Systems Lab
ETH Zürich, Switzerland
rsiegwart@ethz.ch

Stéphane Magnenat
Autonomous Systems Lab
ETH Zürich, Switzerland
stephane@magnenat.net

ABSTRACT

This paper introduces a visual programming language (VPL) for Thymio II, an educational robot. Our VPL is intended for children in primary school and aims at making robotics programming approachable for young children by creating a close correspondence between the icons of the programming language and the design of the robot. Its two modes of operation — basic and advanced — allow children to learn programming at a level suitable for their current skill. Moreover, our VPL provides a live generation of textual code that eases the transition to textual programming for more advanced children. This paper describes the goals and the guiding principles behind the design of our VPL and demonstrates how an iterative development process with evaluations with children resulted in an improved VPL.

Categories and Subject Descriptors

D.1.7 [**Programming Techniques**]: Visual Programming; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces - User-centered design

General Terms

Design, Languages, Human Factors, Experimentation

Keywords

Graphical Programming, Robotics, Early Childhood, Education

1. INTRODUCTION

Programming through robotics has shown its promise in early-child education in recent years [2, 5]. Thymio II is such a robot – designed specifically for children – that has shown its effectiveness in teaching technology to children [7]. This robot can be programmed through the open-source Aseba environment [6]. The latter provides a textual language, which is accessible for teenagers, but too complex for children.

Visual programming has widely been introduced as a higher-level addition to programming languages to lower the entry barrier [4]. Most of these efforts, however, belong to one of two categories: programs that reside entirely in the virtual world [4], or programs that have a physical counterpart but with a relatively abstract connection between hardware and software [1]. Some efforts were made in combining visual and tangible programming with robotics [3], but tighter integration of virtual world presentation and physical world counterpart is still a research question.

We propose a new visual programming language (VPL) for the Thymio II robot, intended for children in primary school. Built on top of Aseba, it renders robotics programming approachable to young children through its close correspondence between its graphical programming elements and the design of the robot (Fig. 1). The VPL has two modes of operation, basic and advanced, to allow children to learn programming at a level appropriate to their development. Visual programs are converted into text programs in real time, easing the transition from visual programming to textual programming for more advanced children.

This paper and demonstration report the initial design, two rounds of design updates resulting from usage observations by children at public workshops, and provides guiding principles for designing such programming environments.

2. DESIGN PRINCIPLES

While designing our VPL for Thymio II, we took inspiration from Resnick and Silverman's guiding design principles [8]. In particular, we considered the following principles:

Lower floor and wide walls: Starting programming with the VPL should be easy for children of various ages. It should hence foster acquisition of new concepts and grow in complexity without compromising its low entry barrier.

Simplicity: The VPL should be as simple as possible without compromising its usability or functionality.

Tinkerability: The VPL should support incremental development of programs and enable children to experiment with new concepts and features.

Support for self- and classroom-learning: The VPL should enable self-learning and support classroom activities through online documentation with minimal reading.

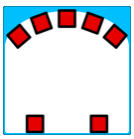
Iterative development: No design is perfect the first time. Every new iteration of the VPL design should be evaluated by its user, i.e., children, and lessons learned from the evaluation should be incorporated into the new design.

Among these, iterative development played a central role in ensuring that our VPL adheres to these design principles. We present three iterations of VPL and demonstrate how the guiding principles and evaluation led to the latest design.

*The work was done while the author was at Autonomous Systems Lab.

IDC 2014
June 17-20 2014
Aarhus, Denmark
www.idc2014.org

(a) Thymio II



(b) An event block

Figure 1: Thymio II and an event block

3. THYMIO II ROBOT

Thymio II [7] is a small differential-drive mobile robot (Fig. 1(a)). It has five infrared (IR) sensors on the front and two on the back to detect obstacles, and two on the bottom to detect ground. There are five capacitive buttons on the top, a three-axis accelerometer, a microphone, an IR sensor for a remote control and a thermometer. The robot also has RGB LEDs on its top and bottom, a circle of eight LEDs on its top, a mono-colored LED next to each sensor, and a sound synthesizer. Thymio II costs 99 CHF (about 80 Euros), and its hardware schematics and software are open source (see `http://thymio.org`).

4. INITIAL DESIGN

4.1 Design

Our VPL is built around one programming construct: event handler. These are built by pairing an event block (triggered by the robot's sensors) and action blocks (setting the robot's actuators). In the first two design phases, a single event and action are composed together into an *event-action pair*. In the current design, multiple actions can be linked to an event. Event and action blocks are designed to closely capture the robot's sensors and actuators. Fig. 1(b) presents an example.

The initial design of our VPL, as shown in Fig. 2(a), has two different modes of operation: basic and advanced. In both modes, textual programs are generated live alongside of the visual programs. This staged programming enables children with various skills and experience to program and operate the robot at the level appropriate for them and aids them to acquire advanced concepts smoothly.

4.1.1 Basic mode

The basic mode contains essential functionalities to operate the robot. It provides five event blocks: buttons, horizontal distance sensors, ground distance sensors, tap detection, and clap detection. It provides four action blocks: motors, top color, circular LEDs, music. Except for tap and clap blocks, all the event blocks and action blocks are individually configurable. Inspired by the traffic light, we use green to indicate that a button is triggered or a sensor detects no obstacle, red to mean stop or a sensor is blocked, and white to mean ignore the state of the button or sensor.

4.1.2 Advanced mode

In advanced mode, we introduce the concept of variable using an *internal state* of the robot. This enables the robot to make a different action upon a given event depending on its state. To set the state, we introduce a state action block, which contains four buttons. Additionally, every event block has four buttons to its right that correspond to four 1-bit state variables. The robot can thus be in 16 different states.

4.1.3 Real-time textual program generation

Visual programs generated in the VPL are converted into text programs in real time. As one adds a new event-action pair to the visual program, the corresponding lines of source code are generated and highlighted. In addition, the child can click on any event-action pair to highlight the corresponding source code. This also means that as the child manipulates a block, the corresponding piece of code also changes live. For instance, when a child activates or deactivates a button in a buttons block, s/he can see how the activation or deactivation gets expressed in textual programming.

4.2 Evaluation

We evaluated the initial design at two annual public events hosted by our university in 2012. We hosted a total of eight 90–minute-long sessions, each with 23 to 25 children aged between 8 and 14 and one teaching assistant for 3 to 5 children. The sessions began with an introduction to the robot and the VPL environment, after which children played with the system freely. The introduction to the robot consisted of an explanation of the sensors and actuators and a demonstration of preprogrammed behaviors. The introduction to the VPL environment consisted of the explanation of event and action blocks and the creation of some simple programs.

We observed the children and noted their questions and confusions to identify strengths and weaknesses of our VPL. In general, all event and action blocks were similarly popular, but differences existed between younger and older children. Younger children preferred to start programming with tap or clap event blocks whereas older children often chose to start programming with buttons or proximity sensors blocks. Most younger children were content programming only in the basic mode, but many older children found the basic mode insufficient. In the end, only about half of the children were introduced to the advanced mode.

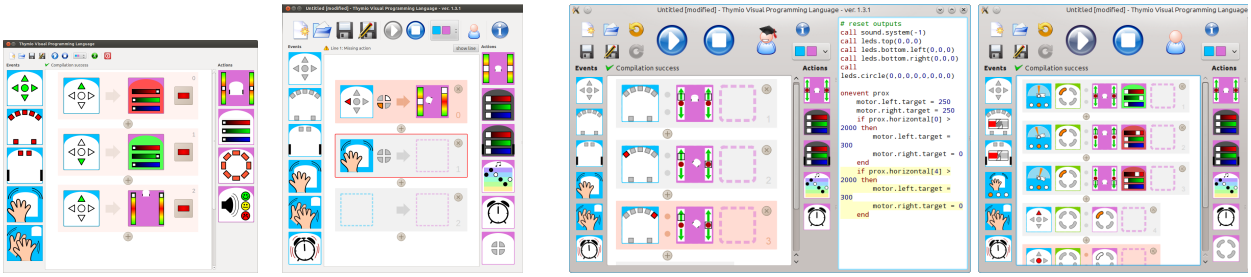
Most children learned about the advanced mode because they demonstrated sufficient understanding of the basic mode; some children, however, learned of the advanced mode by accident, i.e., clicking the advanced mode button unintentionally or noticing their neighbor was using the advanced mode. We had made a design decision to disallow switching back to the basic mode from the advanced mode; this, however, showed to be problematic as not all children could grasp the concept of state/variable. We also noticed that not having any indicator on the robot itself of its current state made difficult to develop its behavior in advanced mode.

A small percentage of children learned to program in the textual language. In teaching them textual programming, the live code generation of our VPL proved to be useful. Children could easily learn the relationship between changes in the visual program and the textual code because highlighting an event-action pair results in highlighting its corresponding textual code. Some children went beyond modifying automatically-generated textual code; five children asked about the timer functionality, a functionality that the textual programming offers but the initial VPL did not, and were able to program these timers with some help from assistants.

5. INTERMEDIATE DESIGN

5.1 Design

Based on the initial evaluation, we made several modifications to our VPL as shown in Fig. 2(b).



(a) Initial, basic mode (b) Intermediate, adv. (c) Current, basic mode (with textual counterpart) and advanced mode

Figure 2: The three iterations of the design of our VPL

5.1.1 *Basic mode*

In basic mode, we changed the sound action block to add more creative possibilities and removed the circular LED action block. We also changed the color scheme of buttons in the event blocks to grey (ignore the sensor), white (detect nothing near the sensor), and red (detect something near the sensor). We choose this color scheme because it maps to what is shown by the LED next to each sensor on the robot. Lastly, we introduced a timer with corresponding event and action blocks. The timer action block starts a timer with a timeout in the range of 0 to 4 seconds, defined by the user by clicking somewhere in the clock face. The timer event block fires when the timer has elapsed.

5.1.2 *Advanced mode*

In advanced mode, we displayed the current state of the robot in the four corner LEDs of its LED circle. The state action can set each of these four bits to 1 or 0, or to leave it as is. For the event, the state buttons act as filters: when a button is gray, the bit is ignored; when it is orange, the bit must be 1 for the event to occur; and when white, the bit must be 0 for the event to occur.

5.2 Evaluation

We evaluated the intermediate design in several occasions (workshops, interactions with children and teachers, etc.), but in particular by observing 40 children, aged 10–15, at a 1.5-hour workshop taking place at our university. The children worked in groups of two or three with one assistant for every 5 to 7 children (twice as many as in the previous workshop). We gave children a set of recommended tasks. Some groups actively played with the robot and the VPL, programming complex behaviors, while other groups struggled to program even simple behaviors such as making the robot move forward. The variability in the performance of the groups seemed to stem from the motivation and interests of the children.

At this session and other events, we observed that the children struggled with the compilation error caused by redundant event-action pairs. Often, children did not realize that their program contained identical pairs. This leads to an error because when the robot has two instances of the same action type associated with a given event block type and parameters, it cannot know which action to execute.

Children were also confused that the robot would not stop unless a motor action with null speed was programmed. The horizontal and ground sensor event blocks were problematic because they can be programmed to cause an event to occur when something is detected or not detected. For the horizontal sensors, an obstacle will cause the IR light to be reflected, while for the ground sensor, at the end of a table, the IR light is no longer reflected. The children simply believed that an event will occur when "something happens" without realizing that in one case the event occurs when light is reflected and in the other when it is no longer reflected.

6. CURRENT DESIGN

6.1 Current design

The evaluation of the intermediate design resulted in the current version of our VPL, as shown in Fig. 2(c).

6.1.1 *Color coding scheme of sensors*

Despite the tight coupling of previous color coding scheme to the robot, many children found the color coding scheme confusing, especially for the ground sensor. Indeed, for the ground sensor to detect a white ground, the color on the block had to be red, and to detect a black ground, the color had to be white. The new version changes this code to black when something is far, and to indicate something is close, to white for the ground sensor and to red for the proximity sensor. As this new scheme captures the physical reality more closely, it would be easier to write programs such as line following and to teach the physics of the sensor.

6.1.2 *Multiple actions per event*

To minimize the errors caused by duplicate event-action pairs, we introduced *event-action sets* that allow multiple action blocks (of different types) to be associated to a given event, and forbid the same event with same parameters to appear twice in the program. In advanced mode, we combined the state buttons associated with events in a *state filter* block, which looks like another block type with a different background color. This improves the consistency of the interface and enables moving and copying of the state filter.

6.1.3 *Updates to blocks*

We updated several blocks. For the sound and timer action blocks, we enabled dragging of notes and the hand, to be consistent with blocks with sliders. For the tap block, which reacts to the accelerometer, we allow it to detect the pitch or the roll angle of the robot in advanced mode. This opens the possibility for interesting behaviors such as avoiding slopes or stopping the robot when it reaches a given inclination. For the proximity and ground sensors event blocks, we allow users to change the thresholds using sliders in advanced mode.

6.1.4 *Other changes*

We made several other additions. To run a program on the robot, children must press the run button; however, children often forget to to so after they modify their program. As automatic loading can break the robot's behavior under execution, we instead introduce a blinking run button to remind children to load their change to the robot. The run button blinks in red when the program is modified and stops blinking when it is pressed. Moreover, as the run and the stop buttons are the most frequently used ones, we increased their sizes to make them more visible. Finally, we introduced the ability to undo/redo changes and to copy blocks.

7. DESIGN DISCUSSION

The current design of our VPL is the result of two iterations of evaluations with children followed by modification. This section discusses how this current design adheres to the principles stated in the introduction and addresses the needs and learning goals of children in primary school.

Low floor and wide walls

The close correspondence between our VPL design and the functionalities of the robot ensures that learning programming with our VPL is easy for children even without any prior knowledge. Our VPL minimizes abstraction to reduce the number of concepts children must learn when starting their programming education. At the same time, the two modes of operation enable children of various skills to learn at their appropriate level; younger children can explore the basic mode while more advanced children can acquire more complex concepts. Our live textual code generation eases the transition from visual programming to textual programming, providing endless possibilities to the most advanced children.

Simplicity

Our VPL ensures simplicity through its clean block design, simple environment, and simple programming construct. The event and action block icons are large and contain only components necessary for parametrization of the underlying event or action. The programming environment has event blocks on the left and action blocks on the right with the central pane for *event-actions sets*. Clicking blocks or dragging them into the center pane is all the children need to do to program. All the relationships between events and actions are encoded in *event-actions sets*, and children can program complex behaviors if they understand this one construct.

Tinkerability

Visual programs can be compiled and run at any time, as long as programs are valid. This means that as children play with our VPL, they can incrementally develop their programs. After adding a new event-actions set, the child can run the program on the robot and see if the robot behaves as s/he expects. If it does, s/he can verify her/his understanding by varying the parameters of the event or the action. If it does not, s/he can question why there is a discrepancy between the expected behavior and the actual behavior. In addition, children can learn what compilation error is when it occurs, i.e., when they create a duplicate event for instance.

Support for self- and classroom-learning

VPL's online documentation provides the possibility of self-learning and classroom activities. The website (`http://thymio.org`), which is available in several languages, provides a 45-page-long tutorial with exercises and answers, and a reference documentation.

8. CONCLUSIONS

We have introduced and demonstrated a visual programming language (VPL) for an educational robot, Thymio II, suited for children in primary school. The goals of our VPL are to make robotics programming approachable for young children and to foster them to deepen their knowledge through exploration and experimentation. The basic mode of our VPL provides children with a gentle introduction to programming and its advanced mode and live textual code generation engage them to go further. The online documentation supports self-learning and provides classroom activities.

The current design of our VPL is the result of two iterations of design and evaluation. We believe that this iterative process enhances the quality of our VPL and its appropriateness to children, and we plan to pursue this process further.

9. ACKNOWLEDGEMENTS

We thank all the children, teachers and co-workers who provided valuable feedback during the development.

10. REFERENCES

[1] D. Benedetelli. *LEGO MINDSTORMS EV3 Laboratory: Build, Program, and Experiment with Five Wicked Cool Robots*. No Starch Press, 2013.

[2] M. U. Bers. *Blocks to robots: Learning with technology in the early childhood classroom*. Teachers College, New York, NY, 2008.

[3] E. Cejka, C. Rogers, and M. Portsmore. Kindergarten robotics: Using robotics to motivate math, science, and engineering literacy in elementary school. *International Journal of Engineering Education*, 22(4):711–722, 2006.

[4] L. P. Flannery, B. Silverman, E. R. Kazakoff, M. U. Bers, P. Bontá, and M. Resnick. Designing scratchjr: Support for early childhood learning through computer programming. In *Proceedings of the 12th International Conference on Interaction Design and Children*, pages 1–10, 2013.

[5] E. Kazakoff and M. Bers. Programming in a robotics context in the kindergarten classroom: The impact on sequencing skills. *Journal of Educational Multimedia and Hypermedia*, 21(4):371–391, 2012.

[6] S. Magnenat, P. Rétornaz, M. Bonani, V. Longchamp, and F. Mondada. ASEBA: A modular architecture for event-based control of complex robots. *IEEE/ASME Transactions on Mechatronics*, pages PP(99) 1–9, 2010.

[7] S. Magnenat, F. Riedo, M. Bonani, and F. Mondada. A programming workshop using the robot "Thymio-II": The effect on the understanding by children. In *IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*, 2012.

[8] M. Resnick and B. Silverman. Some reflections on designing construction kits for kids. In *Proceedings of the 2005 Conference on Interaction Design and Children*, pages 117–122, 2005.