VPL Tutorial Teachers' Guide

Moti Ben-Ari

Version 1.0

First Steps in Robotics with the Thymio Robot and the Aseba/VPL Environment is a tutorial on the *Visual Programming Environment (VPL)* for the *Thymio* educational robot. The purpose of this *Guide* is to assist teachers in using the material in the tutorial in robotics activities with students. Specifically, it includes:

- A description of the concepts that can be studied within each activity.
- Indications of which parts of an activity explore fundamental concepts and algorithms of robotics and which present features that can be explored if time permits.
- Tips for conducting activities with the Thymio and VPL.

Teaching concepts Many concepts have real-world analogs that will be familiar to the students. Nevertheless, it is essential to *explicitly* teach the concepts so that the students will recognize their importance.

Parsons puzzles Parsons puzzles (Chapter 11) are exercises designed so that students construct programs by filling in missing instructions or re-arranging out-of-place instructions, rather than by writing a program from its specification. Parsons puzzles have proved to be very effective in facilitating learning and you are encouraged to design your own puzzles.

From VPL to AESL Once students have mastered programming in the VPL environment they can study how to write programs in the AESL language in the Studio environment. This language gives the user full control over the Thymio robot and enables advanced projects to be constructed. Chapter 19 uses a method called *mediated transfer* to introduce the student to AESL by examining how VPL programs are translated into AESL.

Tips

- Before writing programs with VPL, students can learn about the Thymio robot by exploring the *pre-programmed behaviors* at https://www.thymio.org/en:thymiostarting.
- A summary of the VPL user interface is given in Appendix A and a summary of the event and action blocks is given in Appendix B. These are useful if you or your students forget the meaning of an interface icon or a VPL block.
- Appendix C contains a list of tips for writing VPL programs and for finding and fixing problems. We suggest that you read this carefully before teaching with VPL so that you can guide the students in good practices.

1 Introducing the Thymio

Chapter 1 is an overview of the Thymio robot and the VPL programming environment.

Concepts

- There several general concepts that are presented in this chapter: program, running a program, opening and saving a program.
- The fundamental concepts introduced are *events* and *event handlers*. In VPL, an event handler is called an *event-actions pair* consisting of an event block and one or more action blocks. When the event *occurs* the action is *carried out*.

Tips

- Fully charge the battery before the activities. This is especially important when using the pre-programmed behaviors and when writing programs for a wireless Thymio.
- Students who routinely use computers (including smartphones and tablets) will already be familiar with the concepts of running a program, and opening and saving files.
- It is not necessary to learn all the features of the VPL environment initially. Students must, of course, use the Run and Stop buttons, but since their first programs will be short, New, Open and Save can be skipped during the first session.
- The concepts of event and handler will be familiar, although the terminology may not be known. When playing a game, *click a button* is an event that causes the action *shoot a laser at an alien* to take place. On a smartphone, *touch the green telephone icon* is an event that causes the action *dial a number* to take place. On the Thymio, events include touching a button on the robot and detecting an object by a sensor. Actions include turning on the colored lights and setting the speeds of the motors.

2 Media

The Thymio robot has five touch-sensitive buttons on its top. There are large colored lights on both its top and its bottom. Chapter 2 starts with projects using the buttons and colors, because they can be fun to play with and they facilitate gaining experience with event-actions pairs in VPL. Chapter 6 shows how to use sound with the Thymio: playing notes and sensing a clap or a tap on the robot.

Concepts

- An event can have more than one action associated with it.
- All colors can be produced by combining red, green and blue (RGB).

Tips

- To keep the first programs simple, it is not necessary to introduce the concept of multiple actions for an event at an early stage. However, the need will come up fairly soon, since VPL does not allow two event-actions pairs to have the same event. For example, if you want the event of touching the center button to turn on the top red light and also to start the motors, multiple actions must be used.
- The production of all colors from RGB is not subsequently used in this tutorial on robotics. You can either skip the topic, let the students discover it on their own, or pose projects for the students to work on. The RGB concept is explored in detail in the Wikipedia article RGB color model.
- The Thymio's capabilities for detecting and producing sound are very basic and learning them is optional. The clap and tap events can be difficult to work with if the motors are running, because the motors produce sound that can cause unexpected events.

3 Fundamental robotics concepts

Chapters 3–5 explore the fundamental concepts of autonomous mobile robotics: sensing the environment, deciding how to respond and carrying out actions, especially moving within the environment. The most powerful features of the Thymio are its nine *infrared proximity sensors* (five on the front, two on the back and two on the bottom) that can be used to detect objects, and its motors that enable the robot to move within its environment.

Concepts

- A sensor can cause an event when it *detects* an object (there is a lot of reflected light). Alternatively, a sensor can cause an event when it *does not detect* an object (there is little reflected light).
- The robot uses *differential drive*: the speeds of the left and right motors can be set independently and the robot will turn if the motors are set to different speeds.
- Real robots don't drive straight! Because of slight differences in the components (both when manufactured and as they age) and because the table or floor surface will be uneven, no two robots behave alike. Even running a robot with the same program multiple times will result in inconsistent behavior.
- When an event occurs, the robot must decide what to do. This is called a *control algorithm*; these algorithms are implemented as VPL programs composed of event-actions pairs. Following a line and turning towards an object demonstrate a central concept of robotics: *feedback control*, where to robot achieves a goal by changing its behavior in response to measurements of the environment. Feedback control is essential to obtain correct and consistent behavior.

- *Conjunction*: An event can be defined to occur only if *more than one* sensor detects (or doesn't detect) an object at the same time. It is implemented by selecting more than one sensor *in a single event block* to be white or black. Conjunction means "and"; for example, an event occurs if sensor 1 *and* sensor 2 both detect an object.
- *Disjunction*: An event can be defined to occur if *some* sensor detects (or doesn't detect) an object. It is implemented by selecting sensors in *different* blocks to be white or black. Disjunction means "or"; for example, an event occurs if *either* sensor 1 *or* sensor 2 (*or both*) detects an object.

Tips

- If you work on a table, always include an event-actions pair to stop the robot if it reaches an edge of the table.
- The sensors are very sensitive to how much light is reflected. Students can experiment with various materials—colored tapes, markers, metal or plastic objects—to see how the sensors behave.
- Objects covered with reflecting tape (used as a safety device by joggers and cyclists) can be detected by the sensors at a significantly greater range than ordinary objects.
- Since no two robots are alike programs cannot be copied from one robot to another without checking that the *parameters*, such as the motor speeds, still work.
- The motor speeds can be set in increments of 50 from −500 to 500. You can observe the setting in the text program on the right side of the VPL window.

4 Timers

Although Chapter 7 on timers is short, the concept is important and should not be skipped. Timers are familiar: a microwave oven uses a timer to decide how long to heat the food.

Concepts

- A timer is a clock that is *set* to a specific amount of time (2 minutes to make popcorn). When that amount of time has *expired*, an event occurs (the microwave bell sounds). There are two VPL blocks associated with a timer: an action block to set the timer and an event block used in an event-actions pairs that specifies the actions to be carried out when the timer expires.
- All events in a VPL program are checked at about the same time and all actions associated with each event that occurs are carried out at about the same time. Timers enable actions to occur in a *sequence*. For example, the robot's motors can be run for two seconds and *then* the motors can be stopped.

Tips

- When VPL is opened initially, it is in *basic mode*. By clicking an icon, it enters *advanced mode*. The reason for the two modes is to display a very simple interface to the beginning student and thus to reduce anxiety; when the student gains more experience, the additional features of the advanced mode enable more sophisticated projects to be constructed. Feel free to change the order of topics and to introduce advanced mode earlier or later.
- The setting of the timer duration can be observed in the text program on the right side of the VPL window. The duration is measured in *milliseconds*—thousandths of a second—from 0 to 4000 (0 to 4 seconds) in increments of 250 milliseconds (1/4 of a second). While the timer is electronic and accurate, mechanical components like the motors and wheels are not, so moving forward for 2 seconds will not always bring the robot to the same spot.

5 States

State is a fundamental concept of computer science that is supported in the advanced mode of VPL. Chapters 8 and 9 explore the use of state when writing programs for the Thymio.

Concepts

- The concept of *state* is familiar: A light in a room is *in the state on* or it is *in the state off*. A fan is *in* one of four states: *off*, *low*, *medium*, *high*. A room can be in one of a large numbers of states describing its temperature: $20.5°$, $18.31°$, $25°$. The Thymio robot can be in one of 16 possible states as shown in Figure 8.2 of the tutorial.
- In programs without states, if the event occurs, the corresponding actions are carried out unconditionally. With states, it is possible to place a condition on an event handler: if the event occurs, the corresponding actions are carried out only if the event is *enabled*—the Thymio is in one of a set of states; otherwise, the event is *disabled*—the Thymio is *not* in one of a set of states.

  Enabling and disabling an event is encountered in the real world, in particular, in safety measures: You cannot start a car unless the brake pedal is pushed down (is in the state *down*). Hotel showers will not allow you to run very hot water unless an additional switch is pressed.
- Enabling and disabling events can be used for:
  - *Sequence*. Like timers, state can be used to sequence actions. A timer causes one event to occur after a period of time has passed since a previous event. With states, actions can be performed in a sequence without waiting. Consider the follow event-actions pairs:

    Event 1 -> Actions 1 and set state S1
    Event 2 and in State S1 -> Actions 2

  Occurrences of Event 2 will be ignored unless the Thymio is in State 1, which will be true only if Event 1 occurs. Therefore, Actions 2 associated with Event 2 will occur *after* the Actions 1 associated with Event 1.

  – *Alternative run*. There can be only one event-actions pair associated with an event. Therefore, an occurrence of an event will *always* cause the same set of actions to be carried out. With states, you can specify that an event will cause different sets of actions to be carried out depending on the state. For example, the event of detecting an object by the center sensor can cause the robot to turn left in one state and to turn right in another.

    If you have experience in programming, you will recognize that this is similar to the construct *if Expression then Statement 1 else Statement 2*. In terms of states, an *if* statement is similar to:

        if Event and in States 1 -> Actions 1
        if Event and not in States 1 -> Actions 2

    The condition *not in States 1* is implemented by creating the set of all states that are not in States 1.

- *Memory*. The Thymio *remembers* state it is in; this can be used to store a value. For example, you can count the number of lines of tape that the Thymio encounters as it moves on the floor. In terms of programming languages, the Thymio has four *variables* v1, v2, v3, v4 that can each store two possible values that we can call 0 and 1. Using state blocks you can write an event-actions pair equivalent to the statement:

    if Event and (v2 equals 0) and (v4 equals 1) -> Actions

- *Unary and binary arithmetic*. The state of the Thymio is represented in VPL by four *quarters* in a block that can be either white or orange (or gray to denote *don't care*). These quarters can be used to count 0, 1, 2, 3, 4, depending on which quarter is orange (or none are orange). Alternatively, they can encode the values 0 through 16 in binary as described in Chapter 9.

Tips

- Implementing non-trivial robotics tasks will almost certainly require sequencing and alternative run, so these uses of states are important and should be taught. The material on memory and arithmetic is a central topic of computer science, but it is somewhat advanced and less fundamental for robotics, so it can be considered optional.

- Pay attention to the circle lights on the top of the Thymio since they are your only indication of the current state.

6 Accelerometers

The Thymio has a three-axis accelerometer, of which two axes (left-right, front-back) can be used in VPL in advanced mode.

Concepts

- *Acceleration* is the rate of change of speed. It can be either positive if the speed increases or negative if the speed decreases. When you step on the gas pedal of a car, it accelerates positively. When you step on the brake pedal, the car accelerates negatively (decelerates). When an airplane takes off, the acceleration is high enough so that you feel pressed back into your seat.
- An *accelerometer* measures acceleration.
- The most familiar acceleration is that of *gravity*. The force of attraction of the earth causes us to accelerate towards the center of the earth; otherwise, we would fly off into space as the earth rotates.

Tips

- The Thymio's accelerometers are not sensitive and can reliably measure only the acceleration caused by gravity. They can be used implement game controllers: changing colors and sounds as the Thymio is tilted from left to right or from front to back.
- Projects using accelerometers can be found https://www.thymio.org/en:creations: slope avoidance, weighing scale, pendulum oscillations. The projects were implemented using LEGO bricks and the AESL textual language, but you might be inspired to develop similar VPL projects.

7 Projects

Chapters 12–19 describe additional projects for the Thymio robot and the VPL environment. Here we give an overview of each project in terms of its purpose, relative difficulty and concepts used. Feel free to modify or extend the projects.

- **Braitenberg creatures**. These tasks are well-known and are a lot of fun to implement. Although the projects are very simple, they explore a wide range of interesting behavior that can be implemented primarily with the sensors and the motors.
- **The rabbit and the fox**. This project is somewhat larger than previous ones; it uses the sensors and the motors together with the timer.
- **Reading barcodes**. The Thymio has five sensors on its front that can be used to decode a barcode. Constructing a working program requires experimenting with the

object containing the barcode and the thresholds of the sensors. Therefore, the solution in the archive uses advanced mode so that the thresholds can be set as described in Appendix D. However, you can try to solve it in basic mode.

- **Sweep the floor**. Since real robots don't drive straight, we emphasized the use of events rather than timers to control the motion of the Thymio. This project tries to get the robot to follow a fixed route using just timers and states. Given the unevenness of the floor and the inconsistencies in the motors, you will have to do a lot of experimenting to get it to work reasonably well.

- **Measuring speed**. The task is to measure the amount of time it takes the robot to move over a strip of tape of a fixed length. By dividing the length of the tape by the time, the speed of the robot can be computed. The project requires extensive use of timers and states.

- **Catch the speeders**. This project is rather cool: use the Thymio to implement a police speed trap by measuring the speed of an object that passes in front of the sensors. It requires the use of timers and states.

- **Finite automata** are an abstraction of computers that have many applications. This project is rather advanced when compared with the previous ones. The goal is to read a *tape* consisting of an arbitrary pattern of reflecting and non-reflecting cells, and to change state according to rules programed into the robot. It uses the bottom sensors in an unusual way inspired by the amazing *light-painting project*. One sensor is used to track a line while the other is used to decode the cells.

  A sample page with a track and cells is available in the archive. Instructions on how to print such pages are given in the tutorial, but they require familiarity with the LaTeX document preparation system.

- **Multiple sensor thresholds**. Sensor squares in the sensor event blocks can be either gray (not enabled), white (an event occurs when the value of the sensor is above a threshold), or black (an event occurs when the value of the sensor is below a threshold). This project introduces a fourth possibility: dark gray (an event occurs when the value of the sensor is *between* two thresholds). This feature enables detection of objects at three distances: far, medium, near.