Interactive workshops with MSL robots

Evelien Snel[1], Rob Burgers[2], Jurge van Eijck, Ton Peijnenburg[3]

[1] Elangco – Language, Technology and Education, The Netherlands
[2] Vanderlande, Veghel, The Netherlands
[3] VDL Enabling Technologies Group (VDL ETG), The Netherlands
evelien@elangco.nl

Abstract. The VDL Robot Sports team consists of engineering professionals and participates in RoboCup MSL. In addition to development of robot soccer technology, an important objective is the promotion of robot technology for the next generations of engineers and scientists. We are convinced that all it takes to enthuse boys, and girls, for technology is to let them experience the joy of it. So we developed a way to give them that experience. We often get requests to participate in technology events like Maker Fairs. This is a great opportunity to reach our target audience by offering them an interesting activity that takes less than one hour. We have developed a way to do interactive robot classes that may take as few as one hour and where young students can control our robots using an easy graphical programming language

Keywords: Interactive Workshop, RoboCup MSL robot

1 Introduction

The VDL RobotSports team [1] consists of engineering professional taking on the challenge of RoboCup Mid Size League (RoboCup MSL [2]). The team operates as part of a foundation that has an additional objective to promote technology for future generations of engineers and scientists. The team and foundation find their roots in the Philips Cyber Football Team, which has participated in RoboCup from 2002 up to 2008. The team is often requested to do demonstrations and workshops with their soccer playing robots, e.g. at the annual Eindhoven Maker Faire.

Typically, demonstrations of soccer robots are passive, where the audience watches the robots playing and is given an explanation of the technology. In some cases, members of the audience can play the goalkeeper role and experience how accurate the robots can shoot. In order to further and better engage the audience, we have decided to look for more interaction in the demonstrations. In that context, we have developed a way to make programming the robots accessible to young students.

Interactive workshops are better received than a passive presentation. We want to challenge students with a problem that was interesting enough for them, and yet small enough to be completed within one hour. Computational thinking becomes more

important in modern education and together with programming can be used for the development of 21st century skills [3] and [4].

In Dutch elementary and secondary education, graphical programming environments such as Scratch are used to teach programming skills, which means that many students in our audience will already have experience with the programming environment [5].

2 Aim of the project

We want to promote technology to young people. We are convinced that all it takes to enthuse boys, and girls, for technology is to let them experience the joy of it. We often get requests to participate in technology events like Maker Fairs. At every event we attend the public is very interested in robots and autonomous soccer playing robots sparkle the imagination even more. People cheer when the robots score a goal. But there is a specific category of people we want to get interested: children and adolescents. They are the target audience for our efforts to stimulate interest for technology, because they represent the future. All we could offer them in the past, a few years ago, was a possibility to drive the robot with a joystick, or take the place of the goal keeper. That was entertaining, but it did not teach the children anything about technology.

So what we needed was a way to offer them an activity that makes them think like an engineer, involves a challenge, produces a visible result, and yet takes less than one hour to compete. The idea was to provide a graphical means for programming our soccer playing robots. Rather than designing a completely new graphical programming environment, we have selected an existing programming environment that fits our needs and is already used in primary and secondary education, amongst others in the Netherlands. It is ready to be used by the students with a minimum of instructions.

We want to give the students the experience of controlling the robot via a program; our robots are NOT remote controlled toys, they are autonomous agents. The software we use in official soccer matches is way too complicated to use in a simple workshop for students.

3 Design & implementation

We used the common web service application architecture known as Representation State Transfer (REST) [6]. The main design idea was to use a REST-interface for communication between the graphical programming environment and a robot. This problem can be subdivided into three separate problems:

1. Implement functions to make the robots perform simple movements, without interfering with the built-in software for autonomous behavior.
2. Provide a REST interface for these functions.
3. Create a user interface for calling the REST interface.

3.1 Simple movement tasks and REST interface on the robot

The software architecture in our robots comprises many small tasks that run in a collaborative multitasking mode. We have added a communication task that handles the REST calls, which is implemented via HTTP requests.

By enabling this communication task and disabling most of the tasks that run during a soccer game, we make the robot respond to low-level movement (and kick) commands. The commands are executed with fixed, limited speed and acceleration to prevent serious accidents.

3.2 User interface with REST interface

Thinking of robots in education, the first (oldest) thing that comes to mind is LOGO, designed in 1967 by Wally Feurzeig, Seymour Papert, and Cynthia Solomon [7]. Logo is widely known for its use of turtle graphics, in which commands for movement and drawing produced line or vector graphics, either on screen or with a small robot termed a turtle.

LOGO can be used to teach all computer science concepts, as UC Berkeley lecturer Brian Harvey did in his Computer Science Logo Style trilogy.[8]

A lot has changed since the 1960s and Brian Harvey has moved from LOGO to block programming in a graphical user interface. This is currently being used in programming courses in schools around the world. Examples of these block programming languages are Scratch and Snap!

3.3 Snap!

The web-based Snap! Was developed by Jens Mönig for Windows, OS X and Linux [9] with design ideas and documentation provided by Brian Harvey from University of California, Berkeley and have been used to teach "The Beauty and Joy of Computing" introductory course in computer science (CS) for non-CS-major students [10].

Snap! already offers support for issuing http requests (REST). We only needed

to connect it to our robot in user-friendly way, so that the students can include the robot in their projects. The Snap! interface for the students is in English or Dutch (can easily be changed to other languages).

We simplified the user interface to cater for primary school students. Avoid decimal fractions by using centimeters instead of meters. A forward, backward, left and right move prevent usage of negative numbers. Also degrees are used instead of radians and left and right turn are now separate building blocks instead of using negative numbers for clockwise rotations.

To accommodate different levels of experience, we did not have to “dumb down” the API for interaction with the robots, all that sort of work could easily be handled in Snap! itself.

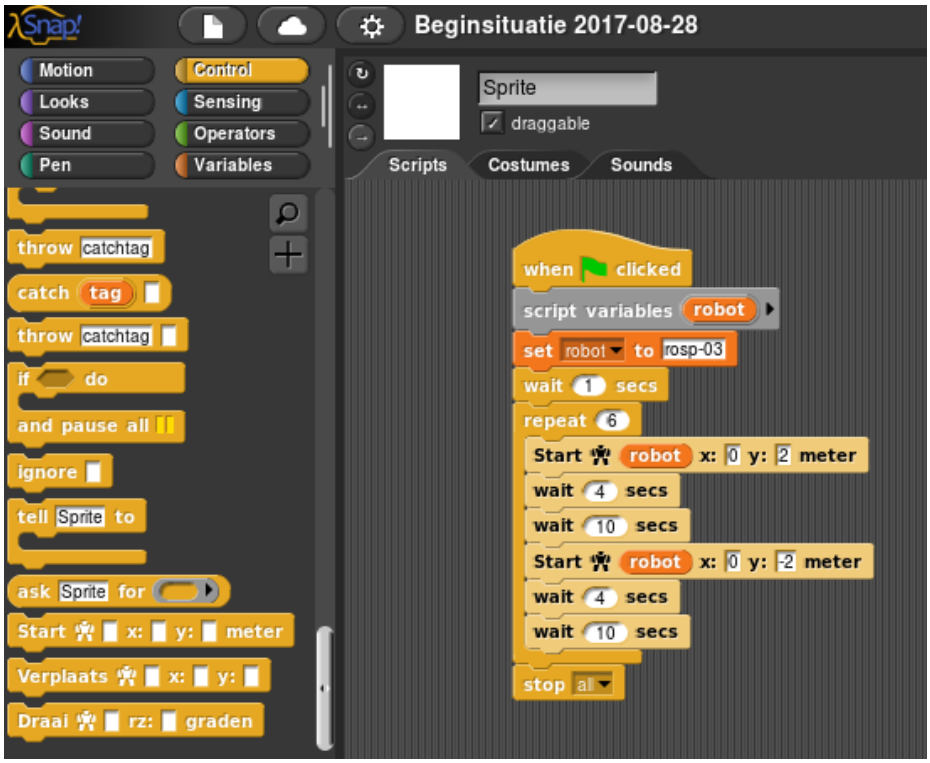

Fig. 1. Snap! interface.

4 Workshops

4.1 Single person - Joystick remote control

In our first year in 2016, kids could remote control the robot by joystick, position it, pick up the ball and kick it – which a lot of kids liked very much. The remote control component was appreciated; moving faster was better, and kicking harder was best.

4.2 Single person - Be the goalie

Before we had our programming interface for students, the best we could do is let the children play as a goal keeper, with the robots taking penalty kicks. This was fun, but that was all there was to it.

4.3 Two teams – Parallel maze tracks

In our second year, we introduced the programming workshop.Each team is provided with a laptop (or a Raspberry Pi), a carpenter's rule, a syllabus, and a robot.

The first workshops were done with a simple maze consisting of two tracks, laid out with black tape on the floor. We managed to successfully run demos in an area of 6x12 meters. One track was the mirror image of the other one, so cheating by copying (and modifying) the results of the other team is almost as difficult as solving the problem itself. Each team is provided with a laptop (or a Raspberry Pi), a carpenter's rule, a syllabus, and a robot.

Instead of teaching a lot of theory, the idea is to make the students explore and to try out their own ideas.

Guided by a compact instruction, the students compose a set of movement commands that direct the robot to move through the maze.

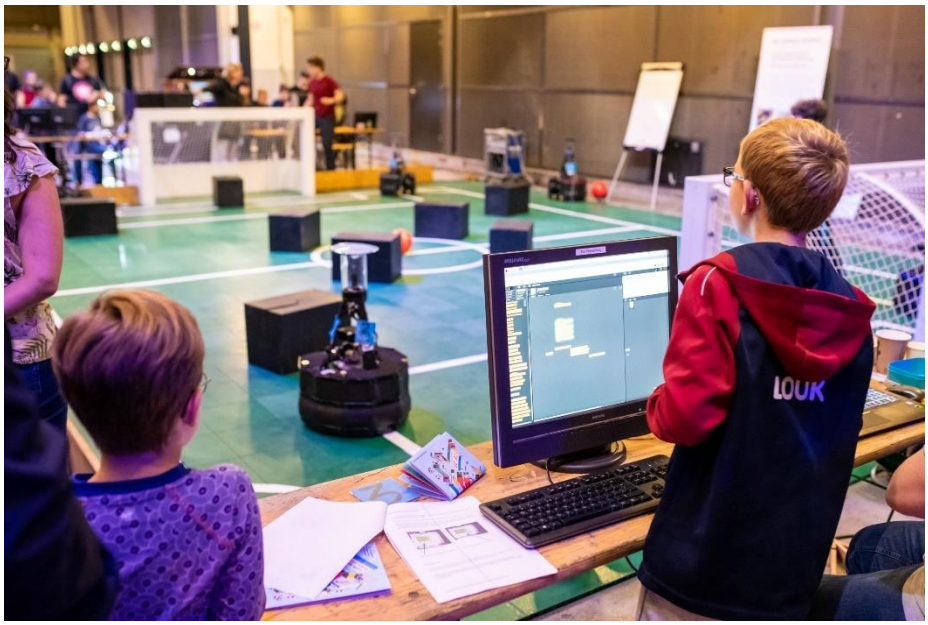

Fig. 2. Student testing his solution for obstacle avoidance

4.4 Four teams – Obstacle avoidance

In 2019 we came up with obstacle avoidance on an indoor soccer field. Four teams could participate at the same time, each starting in one corner of the field. The aim this time was to reach the ball in the center of the field faster than the other teams, avoiding the obstacles (black boxes) we had placed on the field.

We have changed the playing field from the circuits laid out on the floor to our demonstration soccer field, built from plastic tiles and transportable for on-site demonstrations. Using the symmetry of the soccer field, four teams can be accommodated in

the four corner areas of the field. Each team has the quarter of the field to run their robot. The objective is to steer their robot from the corner flag to the ball at field center. Due to black obstacles, they cannot choose a straight path.

The setup not only allows for four teams of five to six students to work in parallel, so a total 24 students, but also offers the opportunity for a match where all teams start their robots at the same time, and compete to get to the ball first. Hitting an obstacle will incur a time penalty.

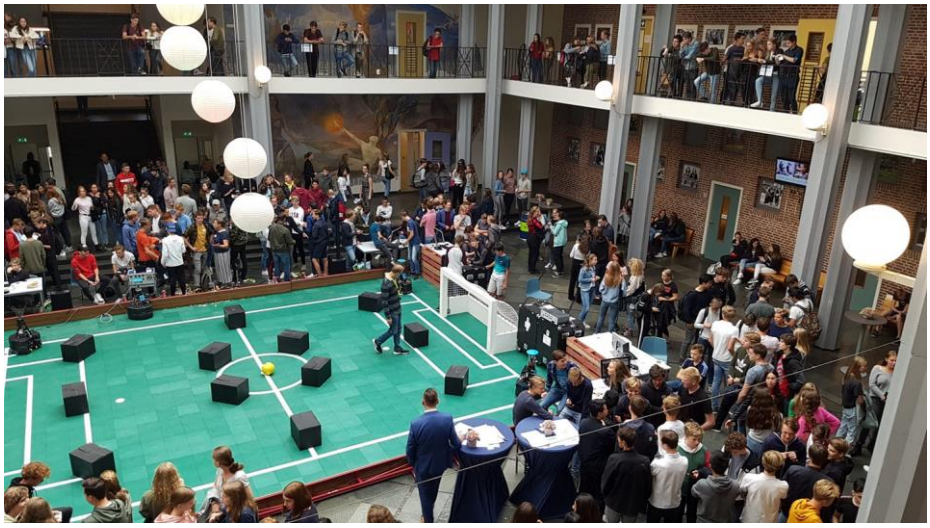

Fig. 3. Workshop at Van Maerlant Lyceum Eindhoven 2019

5 Findings in the workshops

In public events such as a Maker Faire, we need to provide kids with an attractive problem and an attractive environment to catch their attention and keep them involved with the problem for more than a few minutes. There are many other activities at the fair, and many distractions.

In closed events, we have more structure, the students are supposed to pay attention and we can use longer timelines, relying on a classroom-like setting.

5.1 Two teams – Parallel maze tracks

Closed events. The students had 45 minutes to complete their task, which turned out to be enough for all the teams we worked with. Some teams even managed to make the robot go back to where it started from, following the path in the opposite direction.

Maker Faire 2017. Although our simple two-track maze did not stand out well at the Eindhoven Mini Maker Faire, we had good attendance. The attention span was limited to 20 minutes, after which the problem and setting were not attractive enough to

keep attention on the problem. We did realize 1:1 maze competitions on the two parallel tracks. In other workshops that year, we had more structure and better concentration of the participants.

Maker Faire 2018. In the third year, we tried to do the robot programming on soccer demo field. The field stood out, attracted many people for the robot soccer games (2 against 2) but the workshop attendance was less than the previous year. There were two workstations at one end of the field. We had a group of young kids, supporting the workshop and luring their peers into the activity. The maze assignment was not well structured – there were no explicit parallel track lines on the field and the assignment was therefore not clear.

5.2 Four teams – Obstacle avoidance

Maker Faire 2019. In the fourth year, we used the quadruple subdivision of the field, with workstations (Raspberry Pi) at the four corners. Every hour, on the hour, we had a demonstration soccer match and in between we did the workshops. Picking up the ball from the middle, starting in one of the corners with boxes as obstacles proved to be sufficiently interesting. Oftentimes, kids took on the assignment with one of their parents, a sibling or a friend. On average, kids spend about 15 minutes concentrated, trying to program a path for the robot to the ball at the field center. Moving back the robot after a trial run, aligning and adjusting the program were iterated 5 to 10 times. The combination between kids and their parents was interesting, because the kids were explaining block programming using Snap! to their parents. A popular solution for the assignment was to map key presses to motion commands in Snap!, basically building a remote controller. Visiting teachers explained how they were using Scratch at their schools and were jealous of our robots – they just had the turtle.

Event at Van Maerlant lyceum. A school workshop for secondary education was also conducted. In two rounds, four teams of 6 students were given two workstations, a robot and the assignment to program a path to pick up the ball. After a 15 minute class introduction into the robots and the assignment, the teams took to the field and started on their assignment. We had planned for 30 minutes of team work to develop a solution, and then 30 minutes for a test per team and a competitive test with all teams at the same time. We found that intermediate test were required to keep teams on track. At the start of the assignment, teams had to measure the field to determine the distance of the subsequent robot motions. We had only one tape measure available. Some teams complained they had to wait for the tape measure, others simply asked us for the dimension of one tile of the field so they could count themselves. The dual workstations were used by multiple teams to gain time; while one team was executing a robot path, the others were using the second workstation to make adjustments based on their preceding experiment. We arrange for coaches for each of the teams, plus a referee to keep track of time and to call for intermediate trials. In addition, each team had a school teacher as coach; we found that also the teacher got really engaged in solving the task.

Not all the students showed the same amount of concentration. In one of the teams, the boys showed no interest at all, but one girl did not give up. Clearly the intellectual challenge of “solving the puzzle” made her persevere. The robot became her tool for

reaching a perfect result. And when it was time for the final test, she beat all the other teams by reaching the ball first.

6 Future plans and possibilities

The REST interface offers great possibilities for many applications. Http requests do not have to come from the graphical user interface, they may just as will originate from a bash shell script, or a Python program, allowing to adopt the user interface and control side for the workshops to adopt to the capability of the workshop students.

It is also possible to create functions within Snap! that simulate the behavior of the robots. Using that, it would be possible to prepare scripts at home or in the classroom, which can then be used with the real robots in a final event (excursion to our location) at the end of a course.

In the near future we will experiment with adding more of the basic higher level tasks in our software to the interface. Skills like “pick up ball” will greatly increase the activities we can include in the workshop, offering workshop participants the opportunity to experiment with high-level behavior and synthesize robot behavior rather than simple motion sequences.

7 Conclusion

Controlling soccer robots using a REST API from a graphical programming environment provides great opportunities for interactive workshops, attractive for primary and secondary education and with the ability to capture students’ attention to solve a positioning problem using programming. We have participated in several events now and we have seen the enthusiasm we were hoping to create in the eyes of many students. We have seen the intensive concentration and perseverance of children while they are trying to make things work and we have observed their enthusiasm when they succeed. Seeing the smiles on their faces makes us happy too: we have reached our goal and made them experience the joy and beauty of technology.

There is still plenty of room for additional functionality and more interesting workshops in the future.

References

1. Robot Sports Homepage, https://www.robotsports.nl, last accessed 2020/01/30.
2. RoboCup Middle Size League Homepage, https://msl.robocup.org, last accessed 2020/01/30.
3. Gary Ka-Wai Wong & Ho-Yin Cheung (2018) Exploring children’s perceptions of developing twenty-first century skills through computational thinking and programming, Interactive Learning Environments, DOI: 10.1080/10494820.2018.1534245
4. Halil İbrahim Haseski , Ulaş İlic & Ufuk Tuğtekin, (2018) Defining a New 21st Century Skill-Computational Thinking: Concepts and Trends, DOI:10.5539/ies.v11n4p29

5. José Antonio Rodríguez-Martínez, José Antonio González-Calero & José Manuel Sáez-López (2019) Computational thinking and mathematics using Scratch: an experiment with sixth-grade students, Interactive Learning environments, DOI: 10.1080/10494820.2019.1612448
6. Architectural Styles and the Design of Network-based Software Architectures, PhD. Dissertation, R. Fielding, 2000 (htttp://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm)
7. Abelson, Hal; Goodman, Nat; Rudolph, Lee (December 1974). "Logo Manual". Artificial Intelligence Lab, Massachusetts Institute of Technology. Archived from the original on September 11, 2016. Retrieved August 28, 2016.
8. Computer Science Logo Style, Brian Harvey, MIT Press (3 volumes) ISBN 0-262-58148-9, ISBN 0-262-58149-3, ISBN 0-262-58150-7. Available online Archived 2013-07-04 at the Wayback Machine
9. Snap! Homepage, https://snap.berkeley.edu, last accessed 2020/01/30.
10. "UC Berkeley EECS - CS10 : The Beauty and Joy of Computing - Fall 2011". inst.eecs.berkeley.edu. Retrieved 21 May 2017.