Aseba-Challenge: an Open-Source Multiplayer Introduction to Mobile Robots Programming

Stéphane Magnenat[1], Basilio Noris[2], and Francesco Mondada[1]

[1] LSRO laboratory, EPFL, 1015 Lausanne, Switzerland,
stephane at magnenat dot net,
http://mobots.epfl.ch/aseba.html
[2] LASA laboratory, EPFL, 1015 Lausanne, Switzerland

Abstract. We present in this paper a realistic looking robotic survival challenge simulation. Using a model of the e-puck, an open-source mobile robot designed for educational purposes, our simulation allows several players to program the behaviour of e-pucks competing for food. The simulation was tested and well accepted by a hundred children between 7 and 16 years of age. We think that this type of simulations holds a great pedagogical potential because of their inherent fun.

1 Introduction

Teaching robots programming is complicated. Indeed, to be able to successfully program even the simplest behaviour, one has to understand the basics of programming as well as the basics of robotics. The firsts encompass concepts such as variable, execution flow, loops, and so on. The seconds encompass concepts such as sensors, actuators, mobility, perception-action loops, and so on. Yet teaching robots programming, but also teaching programming *through* robots, holds a tremendous pedagogical potential because of the innate human empathy towards animated artifacts, in particular from young people. For example, in our university we use this interest and teach embedded programming to micro-engineering students using robots. We also run a yearly robotic contest.

In this paper we present ASEBA-Challenge, an open-source multiplayer game tailored at teaching mobile robots programming. We also present the result of its deployment in a general public event where it was used by hundred children between 7 and 16 years. ASEBA-Challenge is composed of two parts: a shared 3d arena where simulated robots evolve, and one integrated development environment per player in which they can program and debug their robots. Our goal with this setup is to push down the minimal amount of burden needed to program the robots, and thus to maximize the fun o of the players and the educational value.

ASEBA-Challenge is part of the ASEBA technology, which also works on physical robots of different complexities. In particular, we simulate the e-puck mobile robot which exists physically and the controller can be ported to it with minor changes (the real sensors are less idealistic). Therefore we provide a smooth educational path for people interested in real-world robotics.

2 Related Work

In this section we present some noteworthy approaches that teach programming through mobile agents, physical or virtual. Our aim is not to give an exhaustive survey of the field, but rather to plunge ASEBA-Challenge into context.

The Logo programming language is an early attempt to teach programming with fun [3]. At its root, Logo is a user-friendly dialect of Lisp, but is mostly known for its use in education through its turtle graphics. In this mode, it allows drawing of vector graphics by procedural programming of a virtual turtle. The turtle holds a pen and its movements produce graphics. It is easy to produce nice looking graphics and Logo is motivating for students.

The Lego Mindstorms is probably the easiest way to build physical robots [5]. The Mindstorms consists of a Lego brick enclosing a microcontroller and batteries. It can simultaneously connect to three motors and three sensors out of a large set, including touch, sound, light, ultrasonic, etc. The Mindstorms is an ideal tool for introducing children to robotics, as it allows them to build robots of increasing complexity with bricks they are familiar with. Furthermore, the Mindstorms set provides a graphical programming environment. Unfortunately, being a set of physical devices, it is expensive and is subject to robustness and noise issues.

More recently, the availability of quality game engines including physical simulators, scripting, and display opened new educational possibilities. By allowing children to program worlds they are already familiar with, modding a video games [2] holds a good potential for education. This approach being recent, it still lacks pedagogical maturity: the development tools typically do not provide any debugger, and the games do not provide pedagogical tutorials custom tailored at modding.

Rat's life[3] is a robot programming contest, based on the Webots simulator [7] and the e-puck robot. Like ASEBA-Challenge, it allows players to compete in a virtual arena using a model of the e-puck. Unlike ASEBA-Challenge, the robots must be programmed in C, which is more complex and less user-friendly than our script. Moreover, only two players can compete concurrently and the edit/test cycle of the robot behaviour is much slower. Rat's life is using more standard technologies than ASEBA-Challenge and one can easily build its arena with Lego. Porting is easy, but at the price of a reduced user-friendliness. It also radiates less fun than ASEBA-Challenge, lacking the artistic touch of a video game that is important to attract and retain the attention of players.

Ceebot from Epsitec[4] is one of the closest works to ASEBA-Challenge. In Ceebot, the player also programs a simulated robot in a 3d environment. Ceebot comes with an elaborate tutorial of high educational quality. While Ceebot has a more sophisticated language (including objects, ...) and environments than ASEBA-Challenge, it lacks the multiplayer aspect and the portability to a physical robot.

[3] http://www.ratslife.org
[4] http://www.ceebot.com/ceebot/index-e.php

Each of these approaches brings a creative solution to streamline the learning of programming, yet none of them combine the fast develop/test cycle of an integrated development environment, the social dynamics of a multiplayer game, and the deep motivation inherent to robotics. Aseba-Challenge combines the three.

3 Aseba-Challenge

Aseba-Challenge is a mobile robots simulation game where several robots compete for limited food resources in a closed arena. Each robot starts with a given amount of energy that decreases over time. When the robot feeds, the energy increases, as do its chances of survival. Points are earned by staying alive as long as possible. When the energy of a robot reaches zero, the points earned are halved and its energy is reset to the initial value. When the game starts, the robots are immobile at a random location in the arena.

The player connects to a robot and programs its behaviour using a set of simple instructions and conditional switches. The player can lay down the first blocks of behaviour in a single player environment. Once the basic elements are accomplished, such as endowing the robot with obstacle avoidance abilities, the player can connect to a multiplayer arena to test and fine-tune the performance of the robot against the other players.

3.1 Underlying Technologies

Aseba-Challenge is based on three open-source technologies we have developed: aseba, the e-puck robot, and the Enki simulator.

Aseba is a framework to ease the development of robots controllers. Its core is a tiny virtual machine with embedded debugging logic, small enough to run on any 16 bits microcontroller or better. This virtual machine provides dynamism (loading the program is very fast) and safeness (no script mistake can crash the robot). An integrated development environment interacts with several of those virtual machines concurrently through events, which makes aseba suitable for developing controllers for multi-processors robots [6] or collective robotics experiments. In aseba-Challenge, each robot runs one virtual machine.

The e-puck is an open-source/open-hardware educational differential wheeled robot[5] (Figure 2, right). It consists of a cylinder of about 7 cm in diameter and embeds a large number of sensors: height infrared around its body, three microphones, a three axis accelerometer, and a vga camera. It also provides a loudspeaker and a leds ring. At the time of writing, a fully assembled and tested e-puck is typically available from vendors at the price of 950 CHF.

Enki[6] is an open-source robot simulator that we use to simulate the e-puck. Enki provides collision and limited physics support for robots evolving on a flat

[5] The e-puck mechanical and electrical schematics, blueprints, and source code are freely available at http://www.e-puck.org

[6] http://home.gna.org/enki

surface. It includes a realistic model of the e-puck infrared sensors, as well as a simplified model of its camera.

3.2 Arena

In the development of an environment for mobile robot programming, aesthetic considerations are seldom of paramount importance. However, our goal is to introduce children and young people to robots programming in an entertaining way. For this reason we opted for a realistic looking, yet minimalist environment. The game arena consists of a closed space with four food stations and a number of energy-hungry e-pucks (see Figure 1).

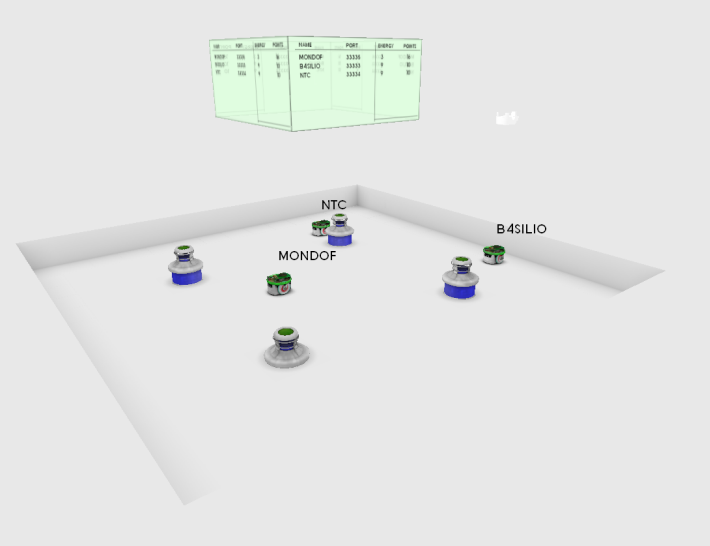

Fig. 1: The ASEBA-Challenge arena. A number of e-pucks searching for the food sources

In developing the graphics and layout of the simulated environment, we wanted to have a clean and easily readable interface while attaining a high level of realism. Moreover we wanted the system to run in real-time on any type of machine. We modeled the main elements of the environment (namely the e-puck robot, the arena, and the food stations) using a commercial 3d modeling software (Newtek Lightwave3d[7]). The e-puck model was constructed from scratch using the technical design files as blueprints and counts 1671 polygons. As we were not aiming for fancy graphical effects but for a believable minimalist environment, we

[7] http://www.lightwave3d.com

opted for a static lighting environment, and computed for each model the radiosity [4] response to the global lighting. Figure 2 shows the e-puck in the different stages of development. Once the color texture and radiosity maps are generated and composed together, we use simple OpenGL 1.1 texture mapping to display them in real-time. The shadow for the e-puck and food stations is added as an alpha-layered polygon with radiosity map attached to the bottom of the model. The whole system counts five 1024x1024 textures (color and shadow maps for the e-puck and food stations, radiosity for the arena). A standard computer can load a score of robots with no apparent drop in performance.

These choices allow ASEBA-Challenge to target a large public (because of reasonable graphics cards requirements) while being attractive enough to retain the attention of children for a long time. Moreover, a realistic looking robot facilitates the identification with a physical robot.

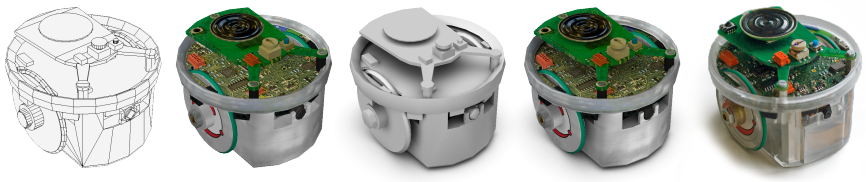

Fig. 2: Development steps for the creation of the e-puck model. Left to right: wireframe of the model; texture map; radiosity map; texture and radiosity maps combined; photo of the real robot.

3.3 Integrated Development Environment

When learning programming, it is important to have a gentle, tolerant, and failsafe environment. In ASEBA-Challenge, players program their robots through an integrated development environment, which provides the following features (Figure 3):

- **Failsafe**. No script mistake can crash neither the environment nor the robot.
- **Syntax highlighting**. The script editor highlights the syntax and colors errors in red. This increases the readability of the scripts.
- **Instant compilation**. The environment recompiles the script while the player is typing it. The result of compilation (success or a description of the error) is displayed below the editor, which permits the correction of errors as soon as they appear.
- **Variables inspection**. The environment lists the variables available on the robot along with their values. The player can update this list in a single click.

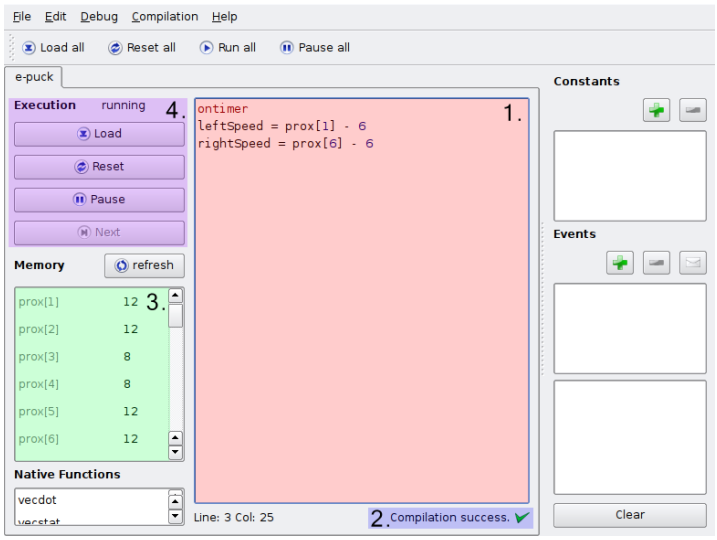

Fig. 3: Screenshot of the integrated development environment. The enclosed elements are the following (clockwise): 1. Editor with syntax highlighting; 2. Message bar from instant compilation; 3. Variables and their values; 4. Debug commands.

– **Debugger**. The environment integrates a debugger. It gives the current execution status, supports continuous execution, step by step, and breakpoints. A right click inside the script editor allows to set or clear a breakpoint on a specific line.

3.4 Programming Language

Players program their robots by writing scripts in a simple language. Syntactically, this language resembles *matlab*. This similarity enables players of ASEBA-Challenge to feel at ease with most scripting languages. Semantically, this language is a simple imperative programming language with a single basic type (16 bit signed integers) and arrays. This simplicity allows players to program behaviours with no prior knowledge of a type system, integers being the most natural type of variables to the general public.

The rationale behind building a language/compiler from scratch is to have the full control over the language syntax and semantic, over the build process, and over the generated bytecode. This allowed us to implement the aforementioned integrated development editor, including the remote debugging of the robot, with relative ease.

We will now take on describing the features of ASEBA script that are useful from an educational point of view[8]:

[8] the complete list of features is available at http://mobots.epfl.ch/aseba.html

– **Comments**. Comments begin with a `#` and terminate at the end of the line.

```
# this is a comment
var b   # another comment
```

– **Variables**. Variables refer either to single scalar values or to arrays of scalar values. The values are comprised between -32768 and 32767, which is the range of 16 bit signed integers. We can access arrays elements using the usual square parenthesis operator; arrays indexes begin at zero.

```
var a
var b = 0
var c[10]
var d[3] = 2, 3, 4
```

– **Expressions** and **assignations**. Expressions allow mathematical computations and are written using the normal mathematical infix syntax. Assignations use the keyword `=` and set the result of the computation of an expression to a scalar variable or to an array element. Aseba provides the operators `+`, `-`, `*`, `/`, `%` (modulo), << (left shift), and >> (right shift). The most precedent operators are `*`, `/`, `%`; followed by `+` and `-`; followed by << and >>. To evaluate an expression in a different order, we can use a pair of parenthesis to group a sub-expression.

```
a = 1 + 1
b = (a - 7) % 5
b = b + d[0]
c[a] = d[a]
```

– **Conditionals**. Aseba provides the `if` conditional and the operators `==`, `!=`, `>`, `>=`, `<`, and `<=`.
The block following the `if` is executed if the condition is true, and the code following the `else` if it is false.

```
if a - b > c[0] then
    c[0] = a
else
    b = 0
end
```

– **Loops**. Two constructs allow the creation of loops: `while` and `for`.
A `while` loop repeatedly executes a block of code as long as the condition is true. The condition is of the same form as the one `if` uses.

```
while i < 10 do
    v = v + i * i
    i = i + 1
end
```

A `for` loop allows a variable to iterate over a range of integers, with an optional step size.

```
for i in 1:10 do
    v = v + i * i
end
```

```
for i in 30:1 step -3 do
    v = v - i * i
end
```

4 Results

We presented ASEBA-Challenge at the EPFL Robotics Festival the 19th of April 2008 (Figure 4). During a whole day, more than a hundred of children participated to the challenge in groups of around 30 people. Although the festival was a general public event, because of its location we had the feeling that a large proportion of the children had at least one parent of engineering profession. The children were of both sexes and aged between 7 and 16 years (avg. 12 years). Children programmed their robots either alone or in pairs on a computer station. A staff of 6 computer scientists introduced and supervised the event giving explanations whenever the need arose. Albeit children initially developed their robots inside their own arena, the best robots were allowed to compete in two arenas projected on the front of the room. All the participants were able to program the basic behaviours into the robots (obstacle avoidance, see Figure 5, left) following instructions and with help from the support staff in case of block, and about half of them managed to implement the more advanced behaviours (going to food sources, see Figure 5, right) by their own means. Three children played with the real e-puck and were able to program it, including advanced sensors such as the accelerometer. A good number of people inquired on the availability of the software.

Fig. 4: ASEBA-Challenge deployment at the EPFL Robotics Festival the 19th of April 2008. Photos: Alain Herzog.

4.1 User-friendliness

Most of the players enjoyed endowing the robots with the ability to navigate the arena and look for food. The players were initially puzzled because the goal was

```
# obstacle avoidance using the
# proximity sensors

ontimer
var distanceToWall = prox[0] +
prox[1] + prox[6] + prox[7]

if distanceToWall < 48 then
    leftSpeed = 10
    rightSpeed = -10
else
    leftSpeed = 10
    rightSpeed = 10
end
```

```
# food sources homing using
# a simple three pixels camera

ontimer
leftSpeed = -10
rightSpeed = -5

if camB[0] > 55 then
    leftSpeed = -10
    rightSpeed = 10
end
if camB[2] > 55 then
    leftSpeed = 10
    rightSpeed = -10
end
if camB[1] > 55 then
    leftSpeed = 10
    rightSpeed = 10
end
```

Fig. 5: Examples of codes children wrote at the EPFL Robotics Festival. Distances to walls are in cm, camera values are in percentage of the color component intensity, and wheel speeds are in cm/s. These units correspond to the simple mode of ASEBA-Challenge. The latter can also be compiled with realistic sensors responses corresponding to the physical e-puck.

not to manually control the robot, but to give it an autonomous behaviour. This problem seemed to be made worst because of the Braitenberg style [1] examples from the tutorial; and switching explanations to an `if condition then action` style of programming helped the players to understand faster. However, the players soon overcame this difficulty and most of them acquired a sort of fondness for their robot; the term pet was mentioned more than once. One frequent remark was that the programming language should be in french[9]. The assistants had to translate commands such as `if`, `else`, and the names of variables for the youngest children. Some children made remarks on the presence of unused panels in the integrated development environment. These panels were designed for the research use of the integrated development environment, they were not necessary for ASEBA-Challenge.

We have also noticed that one of the most frequent difficulty was the abstraction from the robot sensors values to the variables content. To improve the pedagogical experience, we think that we should add a schematic graphical display of the sensors values. Technically, we could achieve this by adding a plugin interface to the integrated development environment.

[9] EPFL is in the french-speaking part of Switzerland

As newcomers entered the contest every hour, we gave them a short talk describing the setup, the robot, and the goal. In overall, people managed to get the basic behaviours working in 45 minutes and the more complex ones in one hour and an half.

4.2 Interaction

During the later part of the game, when players had mastered the basic elements of ASEBA-Challenge, a plethora of interesting behaviours appeared.

Some players started using the robot colored ring to trick other robots into escaping from or following them. As most robots were looking for red and blue objects as source of food, the simpler robots were heavily affected by the behaviour of the naughtier ones. This unleashed a response from the other players who had then to cope both with the environment and the other players, requiring their programs to be more dynamical and complex.

Some children also played with several robots in the arena on their own computer, to implement social games with their robots. For instance, a girl programmed several robots to change their colors when they got close to each other. We think that the interest of children towards behaviours involving multiple robots should be further encouraged, for instance by making a derived game where children could play with several robot each.

5 Conclusion

We presented an open-source challenge game in a mobile robots feeding/survival simulation. The robots simulation is realistic looking and sufficiently easy to understand even for children as young as 7 years of age. The interaction between the robots makes the game stimulating in multiplayer mode.

Moreover, the availability of a physical robot allows further pedagogical experience for interested children. While this robot is more expensive than, for instance, a Lego Mindstorms; its union with a realistic looking simulation environment can expose more children to programming while providing only a small amount of physical devices. This can reduce the overall cost, despite the higher price of the individual robot.

Finally, we observe that children enjoy the nice looking graphics and the thrilling gameplay of ASEBA-Challenge. We think that such games hold a great pedagogical potential because of their inherent fun.

Acknowledgments

We thank the participants and the staff of the robotics festival. In particular: Mara Dalla Valle, Cécile Grivaz, Jacques Paul Grivaz, Solaiman Shokur, Jean-Marc Koller, Philippe Retornaz, Martin Voelkle, and Emmanel Eckard. We also thank the four anonymous reviewers for their insightful comments. Finally, we thank Fanny Riedo for correcting the manuscript.

References

1. Valentino Braitenberg. *Vehicles.* MIT Press, 1984.
2. Magy Seif El-Nasr and Brian K. Smith. Learning through game modding. *Comput. Entertain.*, 4(1):7, 2006.
3. Brian Harvey. *Computer Science Logo Style.* MIT Press, 1997.
4. Henrik Wann Jensen. Global Illumination Using Photon Maps. In *Rendering Techniques '96 (Proceedings of the Seventh Eurographics Workshop on Rendering)*, pages 21–30. Springer-Verlag, 1996.
5. Frank Klassner. A case study of lego mindstorms'^{TM}suitability for artificial intelligence and robotics courses at the college level. In *SIGCSE '02: Proceedings of the 33rd SIGCSE technical symposium on Computer science education*, pages 8–12. ACM, 2002.
6. Stéphane Magnenat, Valentin Longchamp, and Francesco Mondada. Aseba, an event-based middleware for distributed robot control. In *Workshops DVD of International Conference on Intelligent Robots and Systems (IROS)*, 2007.
7. Olivier Michel. Webots: Symbiosis between virtual and real mobile robots. In *VW '98: Proceedings of the First International Conference on Virtual Worlds*, pages 254–263. Springer-Verlag, 1998.