A Gamification Concept for Teaching Swarm Robotics

Heiko Hamann
Institute of Computer Engineering
University of Lübeck
Lübeck, Germany
hamann@iti.uni-luebeck.de

Carlo Pinciroli
Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA, USA
cpinciroli@wpi.edu

Sebastian von Mammen
Games Engineering
Universität Würzburg
Würzburg, Germany
sebastian.von.mammen@uni-wuerzburg.de

September 14, 2018

Abstract

Over the past few years, robots have found their way to the consumer market. With the rise of ubiquitous digitization, the transformative potential of robotics is immense. Yet, it is important to educate a new generation of robotics engineers on researching and engineering mobile robots and also multi-robot systems. Often, students' great intrinsic motivation to learn about this field is quickly shattered due to its inherent multi-faceted challenges. These reach from the need to program robot individuals and to concert collectives to mastering the imperfections of robotics hardware. Accordingly, we present an approach to teaching robotics harnessing the means of engagement and motivation of gamification. We present a concept that builds on the tandem of simulation and hardware trials embedded into a social competitive serious game context.

1 Introduction

With the dawn of robotics flooding the consumer market, we have to find the means to control and direct large numbers of robots [Bock, 2015]. The obvious idea of remotely controlling a mobile unit is quickly challenged when considering many rather than one. Future robots need to exhibit increasing degrees of autonomy. Even more, given their tight integration into our society, robots need to cooperate with each other and ourselves to achieve desirable outcomes. The field of swarm robotics investigates new approaches—in software and hardware—to design and harness the powers of large numbers of autonomous robots that achieve goals unattainable by a single unit [Hamann, 2018].

To involve students into the young, multifaceted discipline of swarm robotics, we propose to utilize the means of gamification and serious games. Gamification means applying playful game elements, for instance badges rewarded to the player, to otherwise non-playful procedures, thus rendering them more attractive [Deterding et al., 2011]. Serious games are understood as "games with a purpose beyond entertainment" [Djaouti et al., 2011]. We propose (1) the gamification of teaching swarm robotics to first-year undergraduate students in computer science, and (2) concrete game elements and serious game mechanics to foster a set of seminal learning targets in the domain of swarm robotics based on established learning theories. Next, we summarize related work in education and serious games. In Sec. 3, we present the state of the art in teaching robotics. Based on our experience, we detail the challenges in teaching swarm robotics in Sec. 4 and outline our proposal for a gamified curriculum in Sec. 5. We outline which frameworks could be used to implement our concept.

2 Related Work

Learning is an innate and rewarding human behavior. Yet, learning can also be a challenge, even when great efforts are invested into effective teaching. Ideally, students are highly motivated, and repeated opportunities for training are provided that give immediate feedback and which also draw from the students' pre-existing knowledge [Ericsson et al., 1993]. Video games, if realized well, are especially suited to address these ideal requirements [McGonigal, 2011]: They are captivating, immerse the player into a constant flow of (repeated) challenges, without underwhelmingly little variation or too difficult tasks leading to frustration. They communicate clear goals and provide immediate and meaningful feedback. The insight that games have meaning "beyond entertainment" [Djaouti et al., 2011] found its way into video games in the 1970s and coined the term "serious games" in the 2000s.

Teaching swarm robotics involves both declarative knowledge such as facts, concepts and algorithms, and procedural knowledge such as soldering on a robot's circuit board [Anderson, 2015]. Since acquiring both types of knowledge may pose equally great challenges and since both aspects need to be properly addressed to succeed, students may be overwhelmed and frustrated. The gap between theory and practice aggravates the situation further, since exhaustive search for errors also needs to consider (unreproducibly) failing hardware components.

Our proposed concept takes these complexities into account and addresses them by following educational approaches, such as the ARCS (Attention, Relevance, Confidence, Satisfaction) model [Keller, 1987]. Founded on expectancy-value theory it revolves around the value the student gains from following through a given learning exercise. The student's attention has to be gained, maintained and directed. The relevance has to be clear in terms of contents and achievements. The student's confidence has to grow by laying out reachable, clear learning targets. The students need to feel satisfaction to be encouraged to follow through and discover by themselves. These requirements coincide with the ingredients of "fun" play experiences—relatedness, competence, and autonomy [Koster, 2013]. Aligned with the learning targets and the audience, we propose to implement this approach by translating the required learning mechanisms to game mechanics, thereby following the according LM-GM (Learning Mechanics-Game Mechanics) Model [Arnab et al., 2015]. For example, short tutorial scenes while playing a game (cascading tutorials as GM) can provide instructional guidance (LM).

3 State of the Art in Teaching Robotics

We briefly summarize the state of the art in teaching robotics from the perspective of mobile and swarm robotics, that is, where large numbers of potentially simple robots interact to accomplish tasks at the collective level. Training programs in robotics typically focus on the design and implementation of robot controllers or the integrative application of robotics to a specific problem, often in a competitive context as to propel the students' engagement. While the latter may require customized hardware designs, there is a selection of robot designs available apt for teaching robotics.

3.1 Robots for Education

Most small mobile robots used in education look similar because an efficient and minimal design is a differential drive combined with proximity sensors. A popular example for teaching swarm robotics across all ages and learning levels is Thymio II [Riedo et al., 2013], a simple, small robot with differential drive, infrared sensors, and some potential for extensions. The strength of Thymio is the considerable software framework consisting of a visual programming language for children of age six and older, Blockly[1] (coding with blocks) for age nine and older, and text programming based on Aseba[2] (tool set for easy programming of robots) for age twelve and older. Another example of a similar design is the e-puck robot. Moradi and Bahri [Moradi and Bahri, 2004] propose to

[1]https://developers.google.com/blockly/
[2]https://www.thymio.org/en:asebausermanual

teach robotics in a hands-on approach with LEGO and find an improved understanding of algorithms and coding methods in students. Despite this variety, a standard course on mobile robotics for Bachelor students typically makes use of a mobile robot in standard differential drive design. In courses on swarm robotics, size and prize of the robot need to be minimized to allow for large robot groups despite limited resources. Again, Thymio is an option here but also the Kilobot [Rubenstein et al., 2012], which is a small swarm robot moving by slip-and-stick motion of legs and with limited capabilities. Most of these simple robots have relatively high robustness and reliability, which helps to limit frustration in students due to hardware failures.

3.2 Robot Simulators and Tools for Education

3.2.1 Programming Languages

To gamify swarm robotics, we have to consider students with different backgrounds. Often students will have had little to no exposure to programming; in other cases, they will have prior programming experience and will seek more complex challenges. An effective software architecture must thus encompass several layers and interfaces, that gradually include more detail as the complexity of the tasks to accomplish increases.

Behavioral specification languages can be categorized into block-based, text-based, and hybrid [Weintrop and Wilensky, 2018]. Block-based educational development environments such as Visual Robotics Platform, based on the well-known Scratch programming language[3], smooth the learning curve for beginners. NetLogo is a widely used text-based educational tool that simplifies the definition of multi-agent behaviors for novice users, although not specifically for robotics applications. Languages such as Python and Lua[4] are ideal for students with prior exposure to programming. Experience in robotic education revealed that robotic middleware, such as Robot Operating System (ROS) [Quigley et al., 2009], can sometimes be inadequate for beginners, due to the large amount of concepts to absorb.

An orthogonal aspect in gamified software environments for swarm robotics is the ease of transferring executable code developed in simulation onto the real robots. This is important in swarm robotics, because the students must manage multiple robots. Currently, the only robotics platform that enables this kind of operation seamlessly is the Kilobot, whose programming API is based on C.

3.2.2 Robot Simulation Environments

Several robot simulators are currently available for robotic education. Among the most widespread, we mention Gazebo, V-REP, and Webots[5]. These simulators offer accurate 3D simulations and integration with a wide set of soft-

[3] http://visualrobotics.org/, https://scratch.mit.edu/

[4] https://www.python.org, https://www.lua.org

[5] http://gazebosim.org/, http://www.coppeliarobotics.com/, https://www.cyberbotics.com/

ware tools and languages. Gazebo's design is intertwined with ROS and offers bindings to several languages, notably Python, and robotic platforms, such as the TurtleBot. Gazebo is arguably the most used tool for robotics education. Robotics educational curricula have also been proposed that use the e-puck along with Webots [Guyot et al., 2011], and the Khepera IV robot along with V-REP [Fabregas et al., 2016]. In all of the above cases, the robotics curricula focus on single-robot navigation rather than on swarm robotics.

3.3 Robot Competitions

There is a tradition of competitions in robotics. Hence, there is potential for using them in education. Competitions can be used to motivate students [Chew et al., 2009]. A drawback is that most of them are focused on research challenges that are beyond what a student team may be able to achieve. Well-known and popular competitions are, for example, RoboCup and the DARPA challenge[6]. Both are characterized by high costs and extreme requirements for teams to enter the competition. Competitions that are easier to enter by amateurs are, for example, the international robotic sailing competition (SailBot)[7] and a various number of robot-sumo competitions (robots try to push each other out of an arena). Sumo competitions are often open for teams of students (e.g., Sumobot competition by FabLab Lübeck[8]). Other examples of competitions organized for school and university students are the VEX robotics competition and the FIRST Robotics Competition[9]. Grandi et al. [Grandi et al., 2014] propose to use LEGO Mindstorms to teach industrial robotics in competitions between teams. An approach combining robotics and gamification is reported by Rursch et al. [Rursch et al., 2010]. Robot competitions that explicitly focus on aspects of swarm robotics are rare but one example is NASA Swarmathon[10] with focus on space exploration [Ackerman et al., 2018]. In hardware three robots need to be coordinated and in simulation also bigger swarm sizes are possible.

4 Challenges

We teach robotics in a hands-on style in several different courses using different platforms. Besides a simple self-built differential drive robot we also use Thymio and the Kilobot [Rubenstein et al., 2012]. We use the self-built robot in a course for first term Bachelor students in combination with a simple graphical editor of finite state machines to navigate mazes. The Thymio is used with the Lua script environment in a Master-level course. In a more advanced course and in student research projects, we use the Thymio with an extended programming environment based on a RaspberryPi extension board and the Python programming

[6]http://www.robocup.org, http://archive.darpa.mil/grandchallenge/

[7]https://www.sailbot.org/

[8]https://sumobot.fablab-luebeck.de

[9]https://www.vexrobotics.com/competition, https://www.firstinspires.org/robotics/frc

[10]http://nasaswarmathon.com/

language. The Kilobot is used in a course that is focused on network aspects rather than an actual introduction to robotics.

We have observed that (1) challenges in hardware design and maintenance are readily accepted by students, whereas (2) they are less willing to invest time and effort into learning the use of simulators. Our intuition is that this bias towards hardware stems from the rare opportunity of working with hardware, the excitement about the robots themselves, as well as generally high expectations in terms of software accessibility known from video games.

Despite their initial excitement, students usually get frustrated once they realize that the robot platforms are not perfectly robust. Due to wear or incorrect handling, sensors fail or programs cannot be uploaded to the robot. Such obstacles add to the already challenging task of programming a specific robot controller, especially since the debugging process needs to consider errors due to hardware ("the robot's fault") and errors due to software ("the student's fault").

A particular challenge of teaching swarm robotics is the collective aspect. When programming a single robot the focus is clear. A student can virtually take the perspective of the robot and think of appropriate behaviors in given situations. In a robot swarm the task is typically defined on the global level, that is the swarm level. Even in research the problem of switching between the individual and the swarm perspective is well-known and challenging [Hamann and Wörn, 2008]. How to program a swarm is, hence, ambiguous and can also lead to frustration.

Even for a fixed robot controller there are many design challenges left for a swarm robotics engineer. To test scalability and robustness of a proposed robot controller, properly designed robot experiments must be conducted in simulation and/or in hardware. Depending on the application, different types of robustness may be relevant. Robots may have to rely on noisy sensor measurements and tests are required to check the robustness of the approach to increased noise levels. Robots may break or partially fail, which requires robustness against dynamic robot group sizes and, for example, faulty messaging between robots. Bottlenecks may emerge due to the effective communication bandwidth or other limited resources such as available space or free memory.

In summary, next to teaching about robotic hardware and its use, programming controllers, designing swarm controllers for desired collective behaviors, the complementary goals that our concept sets out to achieve are as follows. (1) Utilize the hardware appeal to engage and motivate students; (2) Render simulation and other software accessible. (3) Support the distinction between software and hardware errors.

5 Proposed Concept

In this section, we propose a concept to flesh out a viable swarm robotics curriculum based on gamification.

5.1 Bridging the Reality Gap

Both the strong motivational component and the learning targets shaped the idea to teach robotics in an integrative fashion, that is, to develop software and hardware conjointly and incrementally. Both hardware design and robot controllers need to be reflected in a simulation environment. Ideally, there should be only one robot controller program that runs both the simulation and the actual robot. Specifying the controller programs can be realized in text-based code or in visual scripting environments. In analogy, specifying the hardware model can be realized in text-based blueprint specifications or in spatial visual editors (2D or 3D). For both modeling aspects and both text-based and visual approaches, numerous prototypes have been published.

We instruct the students to engage proactively in honing their models, but we also ensure that their hardware implementations do not diverge from their simulation models. Hence, in addition to promoting modeling, we also provide a checkpoint infrastructure to ensure hardware and software are aligned. We define milestones and show differences to the students' code and design but also visualize the simulated results. By retracing the individual steps of a provided sample solution, students could always check their progress and identify potential problems early on. However, depending on their level of expertise, they might want to leap ahead to major milestones that are defined as mandatory by the instructors. The other way around, in case the students explore novel designs and a sample solution is not provided, they can visually debug their design decisions by frequently running the simulation.

5.2 Learning Phases

Students set out to design fully working prototype robots and models from the start but increase their functionality and complexity incrementally, step by step. We are also aware that focusing on programming or on hardware design can be challenging. We propose to split the curriculum into learning phases that always yield a working prototype but may focus on different concrete learning goals. We have identified the following learning phases that we want to incorporate into our curriculum.

1. Familiarizing oneself with a basic robotic system, including the workings and the interrelationship between sensors, actuators and the robot controller.

2. Designing and implementing a robot controller to realize a navigation task of a single mobile robot.

3. Designing and implementing a robot controller to establish collective movement patterns.

4. Optimizing control parameters to improve the achievement of high-level goals.

5.3 Bridging with Badges, Driving with Competition

To address our challenges supported by educational theories, we break the four learning phases into subgoals. Their individual and overall achievements are translated into game elements: Players accumulate points for task completion, leading to game scores giving credit to a player's competence and providing a performance measure. Competition spurs the social relatedness of players. The same is true for badges that are earned when achieving goals. The advantage of badges is that they can convey clear semantics about the concrete achievements (e.g., "Highway of Hell Finalist") and they might be collected to unlock new challenges in the game. Of course, scores and badges can enrich a game side by side.

In the first learning phase, the player might score when successfully activating the robot's differential drive to move forward, to take a turn at the right moment etc. Reaching the goal area is rewarded with a badge. Next to simple activation tasks, simple controllers could be implemented in this phase already that realize simple reactive behaviors and adapt to environmental features. Earning badges unlocks the ability to load the implemented controller to the actual hardware and try it out in real life. Although technically viable, feeding meaningful performance measured from reality back into the simulation requires a lot of ingenuity, including sensor calibration, possibly tracking, and interpreting the data. However, the real-world system could systematically compete against other students' or student teams' systems. Say, a race between two units could take place or the students measure time to completion. Due to its social significance, such a real-world competition should be the primary driver for the curriculum, whereas the scores and badges earned in the simulation would also count towards the students' standing.

The tandem of simulation and real-world deployment would be repeated throughout all learning phases. Whereas simulation becomes increasingly important with the complexity of the robots' tasks, up to the point where simulation results feed into optimization routines to address the great challenge of automatic configuration of swarm robotic systems [von Mammen, 2016].

5.4 Non-linear Gameplay

Autonomy, that is, the freedom to explore on one's own agenda, is important for establishing fun as well. Based on the setup outlined above, there are several ways, and the students/players could set their own goals. For instance, instead of the initial idea of accumulating scores as a secondary means of competitive evaluation, the players could be rewarded with a virtual currency that they can spend on additional sensors or actuators or for tuning the ones they already own—in simulation as well as in reality. An according virtual robot shop could offer just the limited assortment the instructor can provide in reality and offer tuning opportunities that are possible with the available (hardware) parts. As a result, the students could immediately explore the impact of their engineering choices and experience an immediate benefit from the virtual challenges (beyond a working robot design and controller). Offered goods could also include additional robots, the available memory for executing a program on a robot platform, mathematical functions as well as high-level methods that allow the easy specification of certain algorithmic behaviors, or optimizers that can aptly configure robots for a given task and simulation model.

The idea of non-linear play could be taken further still by allowing the students to choose which challenges they want to accept in the simulator and in which order. For instance, they could aim for high-speed racing and pre-script the avoidance of slalom poles and quickly improve their robot's motor and steering abilities or they could give sensory challenges greater priority to increase the robot's autonomy for complex challenges with great rewards that include dynamic environments. These strategic decisions would have a great impact on the students' successes as the overall competition typically runs for a limited period of time.

5.5 Final Grounds with Augmented Reality

At the end of the fourth learning phase, the students would have robot collectives that are customized and optimized to fulfill specific tasks and an infrastructure to adapt their global behaviors to new challenges. They have proven themselves multiple times and the competition could be concluded at this point. However, carrying out one final important competition would encourage all teams to engage until the end of the curriculum. Also, the final competition might yet again surpass the motivation of previous achievements.

We propose a battle royale scenario adapted for swarms, where all the students enter their robot swarms in one final competition and compete against all the others. Clearly, an according battle stage would need to be provided as well as the means of the robots to directly impact each other apart from blocking each others' paths. To this end, we propose to extend the envisioned software/hardware platform by means of player-induced manipulation. For example, the player should be allowed to add robotic units (friends and foes alike) to the battle ground to ensure or challenge the robustness of one's own swarm or an opponent's swarm, respectively. In addition, it should be possible to change individual robots' behaviors based on successful enemy attacks, that is, with a certain probability close-by adversary robots might want to change a robot's configuration. With a rising number of attackers, the odds increase that an attack will succeed and the enemy's swarm would have to prove to be robust (cf. [Tarapore et al., 2015, Irvine and Thompson, 2003]). For changing a swarm member's configuration, the robot's controller should be updated in real-time. An according interface for the player should be offered—the development environment would augment the robots' environment with additional information.

6 Tool Set: Robots in Hardware and Software for Robots

Providing a software-to-hardware workflow is a practical challenge for the envisioned gamified teaching concept. This section proposes a concrete path to its realization.

6.1 Robot Hardware

Our experience shows that most students are highly motivated once the task is embedded into a robot challenge. Complex tasks that may range from designing the robot from scratch to controlling it, are then often accepted and successfully completed. Possibly the above mentioned challenge of distinguishing faults that were caused by the students in their attempt to solve the given task from those caused by faulty hardware is resolved because a properly functioning hardware is part of the students' own responsibility. The students' have to deliver a complete and functional package of robot hardware and robot software. Hence, the challenging task combined with clearly defined responsibilities has a motivating effect on students once they are properly supervised.

We have experience with the above mentioned Sumobot competition by FabLab Lübeck. A team of two students supervised by a PhD student had to design a differentially driven, remote-controlled robot for the Sumobot competition. The students get a clear feedback about their performance via the competition and they are immediately motivated to review and rethink their design and to discuss potential improvements. Arguably this approach can be seen as a way to challenge and encourage high performers.

6.2 Robot Software

We propose to use the recently introduced Buzz programming language [Pinciroli and Beltrame, 2016b]. The main driver in the design of Buzz is to expose to the developer a small, but powerful set of primitives that enables the expression of complex swarm algorithms.

Buzz is also designed as an extension language. Rather than replacing the software stack present on a robot, Buzz is conceived to integrate with it seamlessly, exposing only the relevant aspects of a robot API. This design choice makes it possible to use Buzz with virtually any type of robot, and to add application-specific primitives to the language. The run-time platform is decentralized by design, and offers natively several mechanisms for coordination and communication that work transparently to the developer.

The robots can be heterogeneous in their capabilities and must be able to exchange information and detect each other. In particular, Buzz is based on a form of inter-robot communication called *situated communication* [Støy, 2001]. This communication modality is based on a local message broadcast which is also "situated" because a robot, upon receiving a message, is capable of detecting the position of the sender with respect to its own frame of reference. Situated communication forms the base of a large number of swarm algorithms, including pattern formation, task allocation, aggregation, exploration, and more.

At first sight, the Buzz syntax resembles JavaScript, Python, and Lua. This choice was made to ensure a short learning curve for newcomers. Future work includes Scratch-like visual programming, to enable a more intuitive approach to swarm programmers for beginners. The primitives offered by Buzz include classical constructs, such as variable and function definitions, branches, and loops; and more high-level constructs designed for spatial and network coordination. The `neighbors` construct is a data structure that contains information about the robots in communication range with a certain robot. Through this construct, it is possible to broadcast messages, aggregate local information, filter robots and their associated information, and spatially interact with nearby robots. To propagate information globally across the swarm, Buzz offers the `stigmergy` construct [Pinciroli et al., 2015], a light-weight distributed hash table based on local broadcast and resistant to severe message loss. The `swarm` construct of Buzz allows the programmer to tag robots that respect specific conditions dictated by the developer, and to assign tasks to robots as a function of their tags. Robots can be given multiple tags, and conditions for task assignment can be complex, for example, require a robot to have multiple tags. The ability to tag robots conditionally offers a mixed level of granularity, that enables the developer to express complex swarm algorithms in a comfortable and concise manner [Pinciroli and Beltrame, 2016a]. Two simple algorithms written in Buzz are reported in Fig. 1.

6.3 From Software to Hardware

We propose an integrated workflow for behavior specification that starts in simulation and seamlessly integrates with real robots. Our workflow is based on the Buzz run-time framework and on the ARGoS simulator [Pinciroli et al., 2012]. ARGoS is a fast, general-purpose multi-robot simulator that offers a number of unique features with respect to existing platforms. The most relevant for the definition of a seamless software-to-hardware workflow is the possibility of executing multiple physics engines in parallel in ARGoS. These physics engines can be traditional software components that simulate the dynamics of the 3D models involved; or they can be a piece of software integrated with a motion capture system, such as Vicon, OptiTrack, or a custom solution based on a webcam and OpenCV[11], that feeds positional data to the ARGoS core [Reina et al., 2015]. ARGoS and the Buzz programming language are integrated, thus allowing for a seamless transition from simulation to real robots.

7 Conclusion

We have reviewed the literature on robot hardware, robot simulators, and robot tools for applications in education. We have identified challenges in teaching

[11] `https://www.vicon.com/`, `http://optitrack.com/`, `https://opencv.org/`

```
# Define function to execute upon receiving
# information on the distance to the target
function init() {
  neighbors.listen("dist_to_target",
    function(topic, value, robot_id) {
      dist = math.min(dist,
                      neighbors.get(robot_id).distance +
                        value)})
}
# This function is executed at each time step.
function step() {
  # Broadcast currently known distance to target
  neighbors.broadcast("dist_to_target", dist)
}
```

(a) Gradient formation.

```
# Group identifiers
AERIAL          = 1
TERRESTRIAL     = 2
# Task for aerial robots
function aerial_task() { ... }
# Task for terrestrial robots
function terrestrial_task() { ... }
# Create swarm with robots having 'fly_to'
aerial = swarm.create(AERIAL)
aerial.select(fly_to)
# Create swarm with robots having 'set_wheel_speed'
terrestrial = swarm.create(TERRESTRIAL)
terrestrial.select(set_wheel_speed)
# Assign task to terrestrial robots
terrestrial.exec(terrestrial_task)
# Assign task to aerial robots
aerial.exec(aerial_task)
```

(b) Simple task allocation with heterogeneous robots.

Figure 1: Two examples of Buzz code.

mobile robotics and in particular swarm robotics. Our concept to keep students motivated is based on robot competitions, serious games, and the gamification of robot simulators. These suggestions are based on our long experience in teaching mobile robotics, swarm robotics, and games.

Robotics may have a high impact on industry and society within the next decades. Hence, it is essential to motivate students to study robotics-related fields and to keep them motivated in hands-on approaches to robotics education.

References

Sarah M. Ackerman, G. Matthew Fricke, Joshua P. Hecker, Kastro M. Hamed, Samantha R. Fowler, Antonio D Griego, Jarett C Jones, J. Jake Nichol, Kurt W. Leucht, and Melanie E. Moses. The swarmathon: An autonomous swarm robotics competition. *arXiv preprint arXiv:1805.08320*, 2018.

John R Anderson. *Cognitive Psychology and Its Implications*. Worth Publishers, 8th edition, 2015.

Sylvester Arnab, Theodore Lim, Maira B Carvalho, Francesco Bellotti, Sara Freitas, Sandy Louchart, Neil Suttie, Riccardo Berta, and Alessandro De Gloria. Mapping learning and game mechanics for serious games analysis. *British Journal of Educational Technology*, 46(2):391–411, 2015.

Thomas Bock. The future of construction automation: Technological disruption and the upcoming ubiquity of robotics. *Automation in Construction*, 59:113–121, 2015.

Moi-Tin Chew, S Demidenko, C Messom, and G Sen Gupta. Robotics competitions in engineering eduction. In *4th International Conference on Autonomous Robots and Agents (ICARA)*, pages 624–627. IEEE, 2009.

Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. From game design elements to gamefulness: defining gamification. In *Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments*, pages 9–15. ACM, 2011.

Damien Djaouti, Julian Alvarez, Jean-Pierre Jessel, and Olivier Rampnoux. Origins of serious games. In *Serious games and edutainment applications*, pages 25–43. Springer, 2011.

K Anders Ericsson, Ralf Th Krampe, and Clemens Tesch-Römer. The role of deliberate practice in the acquisition of expert performance. *Psychological Review*, 100(3):363–406, 1993.

E. Fabregas, G. Farias, E. Peralta, H. Vargas, and S. Dormido. Teaching control in mobile robotics with v-rep and a khepera iv library. In *2016 IEEE Conference on Control Applications (CCA)*, pages 821–826, 2016.

Raffaele Grandi, Riccardo Falconi, and Claudio Melchiorri. Robotic competitions: Teaching robotics and real-time programming with lego mindstorms. *IFAC Proceedings Volumes*, 47(3):10598 – 10603, 2014. ISSN 1474-6670. doi: https://doi.org/10.3182/20140824-6-ZA-1003.00222. 19th IFAC World Congress.

Luc Guyot, Nicolas Heiniger, Olivier Michel, and Fabien Rohrer. Teaching robotics with an open curriculum based on the e-puck robot, simulations and competitions. In Roland Stelzer and Karim Jafarmadar, editors, *Proc. of 2nd Int. Conf. on Robotics in Education (RiE)*, pages 53–58. INNOC, 2011.

Heiko Hamann. *Swarm Robotics: A Formal Approach*. Springer, 2018.

Heiko Hamann and Heinz Wörn. A framework of space-time continuous models for algorithm design in swarm robotics. *Swarm Intelligence*, 2(2-4):209–239, October 2008. URL http://dx.doi.org/10.1007/s11721-008-0015-3.

Cynthia E. Irvine and Michael Thompson. Teaching objectives of a simulation game for computer security. Technical report, Naval Postgraduate School, Monterey, CA, USA, 2003.

John M Keller. Development and use of the arcs model of instructional design. *Journal of instructional development*, 10(3):2, 1987.

Raph Koster. *Theory of fun for game design*. O'Reilly Media, Inc., 2013.

Jane McGonigal. *Reality is broken: Why games make us better and how they can change the world*. Penguin, 2011.

Hadi Moradi and Alireza Bahri. The use of "tell me, show me and let me do it" in teaching robotics. In *Proceedings of the AAAI 2004*, pages 160–164, 2004.

Carlo Pinciroli and Giovanni Beltrame. Swarm-oriented programming of distributed robot networks. *IEEE Computer*, 49(12):32–41, December 2016a.

Carlo Pinciroli and Giovanni Beltrame. Buzz: An extensible programming language for heterogeneous swarm robotics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2016)*, pages 3794–3800, Los Alamitos, CA, October 2016b. IEEE Computer Society Press.

Carlo Pinciroli, Adam Lee-Brown, and Giovanni Beltrame. A tuple space for data sharing in robot swarms. In *9th EAI International Conference on Bio-inspired Information and Communications Technologies (BICT 2015)*. ACM Digital Library, 2015.

Carlo Pinciroli et al. ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, 6(4):271–295, 2012. ISSN 1935-3812. doi: 10.1007/s11721-012-0072-5. URL http://dx.doi.org/10.1007/s11721-012-0072-5.

Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. ROS: an open-source robot operating system. In *ICRA workshop on open source software*, 2009.

Andreagiovanni Reina, Mattia Salvaro, Gianpiero Francesca, Lorenzo Garattoni, Carlo Pinciroli, Marco Dorigo, and Mauro Birattari. Augmented reality for robots: virtual sensing technology applied to a swarm of e-pucks. In *2015 NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2015))*, pages 1–6. IEEE Computer Society Press, Los Alamitos, CA, 2015. Paper ID sB_p3.

Fanny Riedo, Morgane Chevalier, Stephane Magnenat, and Francesco Mondada. Thymio II, a robot that grows wiser with children. In *IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*, pages 187–193. IEEE, 2013.

Michael Rubenstein, Christian Ahler, and Radhika Nagpal. Kilobot: A low cost scalable robot system for collective behaviors. In *IEEE International Conference on Robotics and Automation (ICRA 2012)*, pages 3293–3298, 2012. doi: 10.1109/ICRA.2012.6224638.

Julie A Rursch, Andy Luse, and Doug Jacobson. IT-adventures: A program to spark IT interest in high school students using inquiry-based learning with cyber defense, game design, and robotics. *IEEE Transactions on Education*, 53(1):71–79, 2010.

Kasper Støy. Using situated communication in distributed autonomous mobile robots. In *Proceedings of the 7th Scandinavian Conference on Artificial Intelligence*, pages 44–52. IOS Press, 2001.

Danesh Tarapore, Pedro U Lima, Jorge Carneiro, and Anders Lyhne Christensen. To err is robotic, to tolerate immunological: fault detection in multirobot systems. *Bioinspiration & biomimetics*, 10(1):016014, 2015.

Sebastian von Mammen. Self-organisation in games, games on self-organisation. In *Games and Virtual Worlds for Serious Applications (VS-Games), 8th International Conference on*, pages 1–8. IEEE, 2016.

D. Weintrop and U. Wilensky. How block-based, text-based, and hybrid block-/text modalities shape novice programming practices. *International Journal of Child-Computer Interaction*, 2018.