BACHELOR THESIS PROPOSAL: SIMULATING LEGO MINDSTORMS TO FACILITATE TEACHING PROGRAMMING TO SCHOOL STUDENTS

TORSTEN KAMMER

1. Introduction

The go4IT! project teaches school students the basics of programming in workshops [1] using the Lego Mindstorms NXT. This hardware allows users to build a wide variety of robots and offers (among others) a C-like programming language to develop software for them. It has proven quite popular in teaching environments to introduce the basics of programming and robotics [4]. At the moment, workshop participants are generally not able to learn and try out more after the workshop ended due to limited hardware availability. This restricts the usefulness of the project, because if the participants would likely be able to learn more if they had more time to experiment with the robots.

To solve this problem, the goal of this bachelor thesis is to develop a simulator that can execute compiled applications for this platform. This will allow the users to develop software using the same tools as for the actual device and then allows testing them without access to the hardware, both at school and at home.

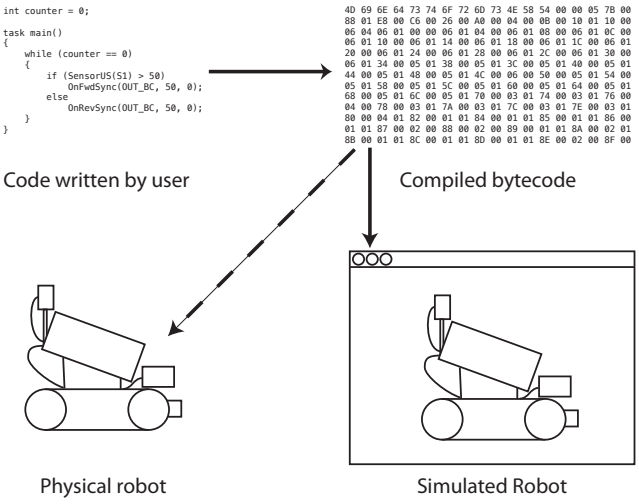


Figure 1. Overview of the development process for a physical robot vs. the simulator.

To enable this, there is no need for the simulation to be fully physically accurate or to provide an exact representation of the simulated hardware. What is required is that programming is as similar to reality as possible, and that the simulator is easy to use even for novice computer users.

Additionally, the simulator will be able to communicate with others over a network. This means that a number of users with their own computers will be able to test their robots together in a shared virtual environment.

The main goal will be to support Microsoft Windows, but through the use of cross-platform solutions like OpenGL and SDL, support for other platforms will be easy to add later.

2. Related Works

In order to develop embedded systems, it is often essential to have good simulation tools. Testing new software on actual hardware can be difficult, for example because the hardware is not available, is too expensive to acquire for every developer, or because incorrect software can damage the hardware. When developing new systems, the hardware may not even be available yet. Simulating the target hardware on a computer allows developers to write and debug code without any of these disadvantages [7].

Simulation environments can be set up in a variety of ways, e.g. simulating the underlying hardware directly (both original application software and operating system), simulating the operating system (using original application software) or simulating the operating system programming interfaces (using original application source code). It is often necessary to simulate both the embedded system and the environment it interacts with. However, it may be appropriate to simulate only a subset of the hardwares capabilities, or to simplify the simulation of the environment, depending on the goals of the project. It is also desirable to have a virtual view of the simulated hardware, which makes it obvious whether the new software works as intended.

There is currently no simulator available that can model the Lego Mindstorms NXT system. Simulators are available for similar systems, such as the previous generation Lego Mindstorms system [8]. However, they are programmed using different programming languages, and are hence not a useful substitute.

3. Objectives

The simulator will be used as a teaching tool, so the features will be specifically tailored to this environment.

- Only a subset of the features of the Lego Mindstorms set will be supported. In particular, only a limited number of robots will be included (with some configuration options), and the simulator will not be able to execute all valid programs. In particular, the programs created in the go4IT workshops will be supported.
- Performance is not a key concern, because the Lego Mindstorms NXT controller has a clock rate of 48 Mhz, while typical personal computers have significantly more than 1 Ghz, i.e. more than twenty times the speed.
- The environment will be simple, but easy to edit, so that users can create a variety of scenarios.
- The physical simulation will not be highly realistic, only enough to appear plausible.
- Network collaboration will be restricted to LAN environments. There will be no optimization for low-bandwidth or high-latency situations.

To achieve this, the architecture of the simulator will follow the standard Model-View-Controller [5] pattern, as illustrated in Figure 2.

The most important objectives arising from this are:

3.1. **Simulating compiled code.** Programs for the NXT 2.0 are compiled to a bytecode format, which is then executed by the controller. A simulator will have to be able to execute at least a subset of this bytecode in order to run programs. The simplest way is to write an interpreter that executes each instruction independently. This does not offer optimal performance, but should be enough for the expected programs.

3.2. **Creating a test environment.** To be able to test the code, it is necessary to have a virtual test environment that offers great flexibility but is not too complicated. It should be possible to edit this environment very easily, so that a wide number of test scenarios can be set up quickly and in an intuitive way. To ensure this, the environment will be based on a grid of blocks. Each block can either form a wall or a floor tile. As the hardware has light sensing capabilities, users can also alter the color of a block.

3.3. **Developing a simulation.** The main goal will be to have a simulation of a robot within the test environment created above, with the robot being controlled by the code the user writes (using the intermediate step of the bytecode). The simulation will not be physically accurate, as the goal is only to show than an algorithm works in principle. Its results will be used to generate data for the virtual sensors of the robot, which will be fed back into the simulated program. While the robot itself will remain in a single configuration, it will be possible to position the sensors in a variety of ways, so that the robot can be adapted for different scenarios.

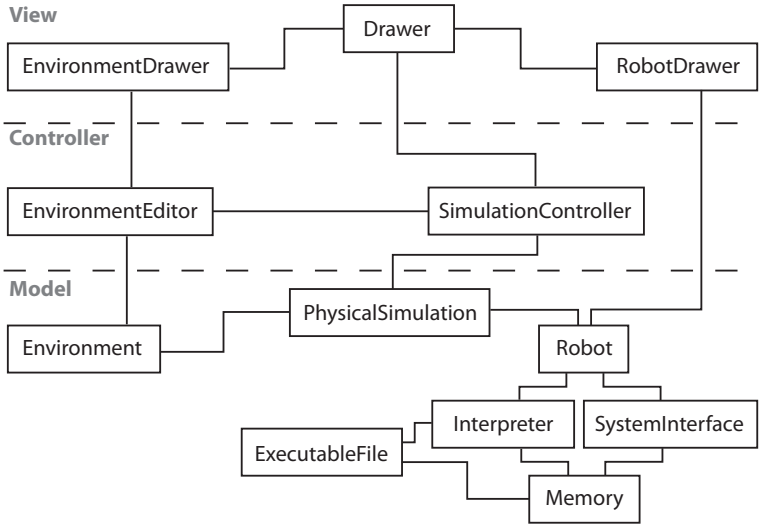


FIGURE 2. Suggested overview over the architecture of the simulator. The network subsystem is excluded for clarification.

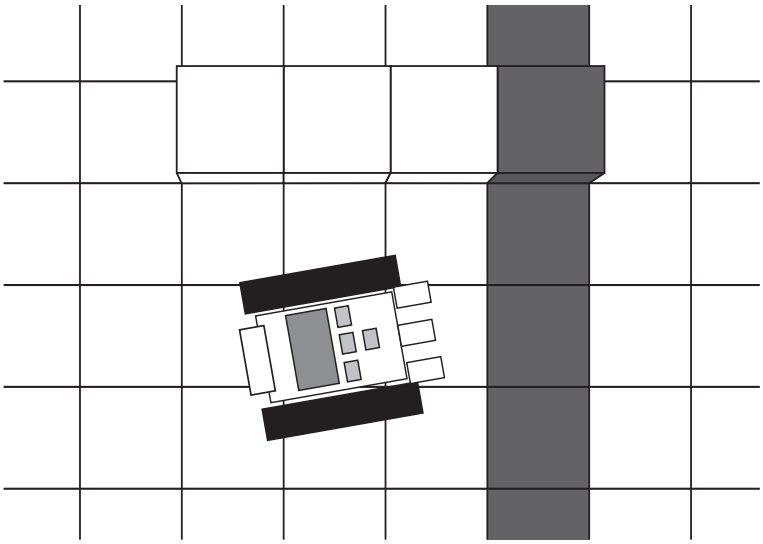


FIGURE 3. Mockup of the simulation environment.

To provide for a maximum level of cross-platform compatibility, the simulation will use OpenGL [6] for graphics, SDL [3] for interfacing with the window system and OpenAL [2] for audio output. All three are libraries suitable for developing for a variety of platforms, including Microsoft Windows [9].

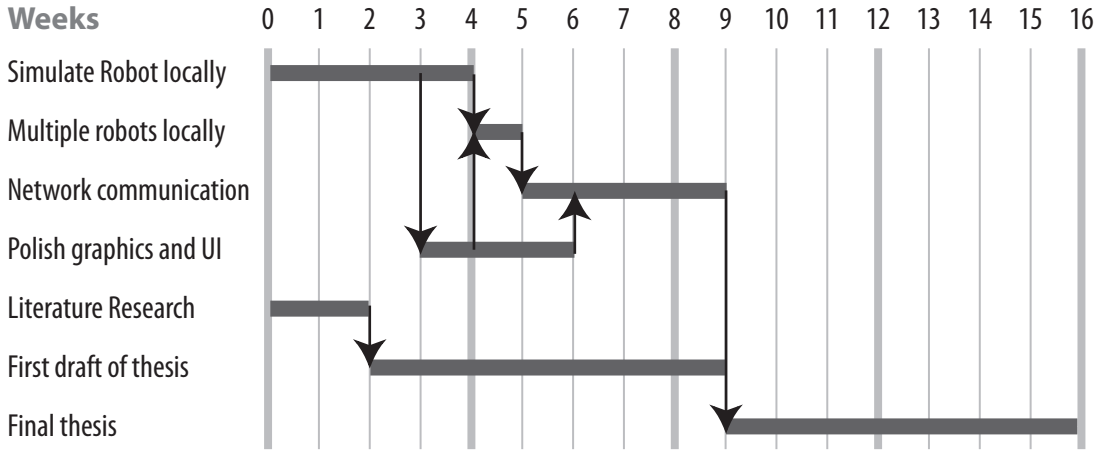
3.4. **Implementing a network layer.** It should be possible to have different robots operate in the same virtual environment, so that users can compare their programs. To ensure this, a network layer is necessary. To reduce development effort, I plan to implement a client-server model, where all robots are simulated on a single computer,

and the other computers connected will simply display the results. The goal is to have a solution suitable for local simulations only, so optimizing for low-speed or high-latency connections is not as important.

3.5. **Integration with BricxCC.** Since the final simulator will be used by students with little technological proficiency, using the simulator has to be as simple as reasonably possible. A way to achieve this would be to modify the BricxCC development environment used in the workshop, so that it will communicate with the simulator instead of an actual physical robot.

4. Schedule

The allocation of time to the various parts of the project is as follows:



5. Contact

Torsten Kammer
torsten.kammer@rwth-aachen.de
+49 170 5546163
http://ferroequinologist.de/

References

[1] go4it! project homepage. http://lehramt.informatik.rwth-aachen.de/go4it , last accessed on April 7th, 2010.
[2] OpenAL homepage. http://www.openal.org/, last accessed on April 7th, 2010.
[3] SDL homepage. http://www.libsdl.org/, last accessed on April 7th, 2010.
[4] D. Eggert. Using the lego mindstorms nxt robot kit in an introduction to c programming .... *Journal of Computing Sciences in Colleges*, Jan 2009.
[5] P. Sauter, G. Vögler, G. Specht, and T. Flor. A model–view–controller extension for pervasive multi-client user interfaces. *Personal and Ubiquitous ...*, Jan 2005.
[6] D. Shreiner, M. Woo, J. Neider, and T. Davis. *OpenGL Programming Guide*. Addison-Wesley, 2004.
[7] U. Siems, C. Herwig, and T. Röfer. *SimRobot, ein System zur Simulation sensorbestückter Agenten in einer dreidimensionalen Umwelt*. Number 1/94 in ZKW Bericht. Zentrum für Kognitionswissenschaften. Universität Bremen, 1994.
[8] G. Theidig, J. Brding, and U. Petersen, editors. *Roberta - Der Simulator RobertaSim*. Fraunhofer IRB Verlag, St. Augustin, 2006.
[9] T. Yu. Chess gaming and graphics using open-source tools. *International Conference on Computing*, Jan 2009.