to School Students

Torsten Kammer[1], Philipp Brauner[2], Thiemo Leonhardt[1],
and Ulrik Schroeder[1]

[1] RWTH Aachen University, Lehr- und Forschungsgebiet Informatik
Ahornstr. 55, 52074 Aachen, Germany
{torsten.kammer,thiemo.leonhardt,ulrik.schroeder}@rwth-aache
[2] RWTH Aachen University, Human Technology Centre
Theaterplatz 14, 52056 Aachen,
brauner@humtec.rwth-aachen.de

Abstract. Programmable robots like Lego Mindstorms have proven be an effective mediator to teach computer programming to school cl dren. Therefore several projects that aim at increasing the interest computer programming and computer science in general use robots a cornerstone in their course concepts. Handing out robotic kits to school students who have participated in the courses is not feasib thus the learning content cannot be repeated and enhanced at hor We developed a flexible multi-user simulation environment for LEG Mindstorms NXT robots which is closely integrated into our pedag ical teaching scenarios. User tests show that this environment can successfully used to increase the long-term outreach of our courses.

Keywords: Simulation Based Learning, CS0, Simulator, LEGO Mi storms, NXT.

1 Introduction

Despite the economic crisis, demand for IT-professionals continues to b Therefore, one goal to achieve is to optimize teaching at schools in qu quantity in order to improve young peoples interest in science, techn gineering and mathematics (STEM). The PISA study revealed a hi in German school education especially in international comparison. Tl mastery of basic skills due to lack of references to applications and ne of teaching has been criticized, as well as rather static teaching scena other aspect is the low number of female students in STEM subjects in school, girls show less interest in topics of computer science in Gern

To increase the participation in STEM in general and the partici women in particular a lot of projects have been initiated. One popula is to use robots as a tool to increase interest in computer programn computer science [6,9,5]. In a controlled experiment we found that prog

C. Delgado Kloos et al. (Eds.): EC-TEL 2011, LNCS 6964, pp. 196–209, 2011.

Therefore we implemented a series of robotic workshops mainly fo the age of eleven to twelve. These workshops last two days and are ca in local schools instead of the regular lessons. The students learn t of computer programming using Lego Mindstorms NXT robots. This allows users to build a wide variety of robots and offers (among other like programming language NXC (Not Exactly C) to develop software Our course concept has run successfully for two years, and approxima school students have participated in our robotic courses. Our evaluati that students like the courses and that their interest in STEM in ge computer science in particular has risen after participation in a works

However, one drawback of our course concept is the limited susta As each robotic set costs roughly 350 Euros, most participants do not privately. Thus participants usually cannot repeat the subject matter ally or deepen their programming knowledge on their own after the has ended. This limits the benefit of the project, because the participa probably learn more if they had more time to experiment with the r overcome this problem we developed a simulator that is closely integr our course concept and can easily be used by individual students or students after the workshop has ended.

The simulator provides a three dimensional view on a standard LEG storms NXT robot and its environment. The simulated robot model ca ified by adding or removing sensors or by changing the positions of th The environment is easily reconfigurable with walls and arbitrarily sha tiles so that a magnitude of learning scenarios can be created. The simu be downloaded for free from our InfoSphere website[1].

We focused on the learning experience with regard to computer scien tion. Thus we didnt provide a physically accurate simulation of the envi the robot and its components, but rather an abstraction as learning t ever, the simulated robot resembles the behavior of a real robot ver Additionally, the simulator is able to communicate with other instanc simulation environment over a network. This allows users to implemen their robot programs collaboratively in a shared virtual environment collaborative scenarios of our go4IT!-workshops.

The simulator directly executes code compiled for the NXC platfo allows users to develop software using the same tools as used in the w for the actual device and test the software without access to the both at school and at home. As the simulator executes native code, it be used with arbitrary third party tools like a programming environn programming language different from the ones we used in our courses.

[1] http://schuelerlabor.informatik.rwth-aachen.de/simulator

move on wheels and can utilize various sensors. The students have bee with LEGO blocks since kindergarten, so there are no problems to buil ing robot. After that the students actively explore the first simple com control the robots movement (forward, backward, different speeds, squa stop for obstacles or at certain distances) and to write corresponding in NXC. At the end of the first day the robots control the touch sensor self-composed and programmed songs. These goals are usually achiev participants.

During the second day students learn to control additional senso ultrasonic and light sensors). Thus, students come up with more advan they want to implement. Potential projects are discussed by the pa and are implemented mostly without help of the tutors. Individual so a team are presented to all students and successful students are enco help the others as experts in their problem-solving. At the end of t day, all participants work on a common task for the whole group such performances and choreographies for interacting robots. In this case, of all teams are involved in the common performance.

The workshop ends with the presentation of this team performanc of classmates, teachers, parents, or sometimes the invited press. The gain recognition for their work through the feedback of the audienc end of the workshop all students have successfully solved tasks inde and developed programs to let a robot perform a certain task. The an solution of complex problems enhance the self-efficacy of the studen *robot does what I want!"*).

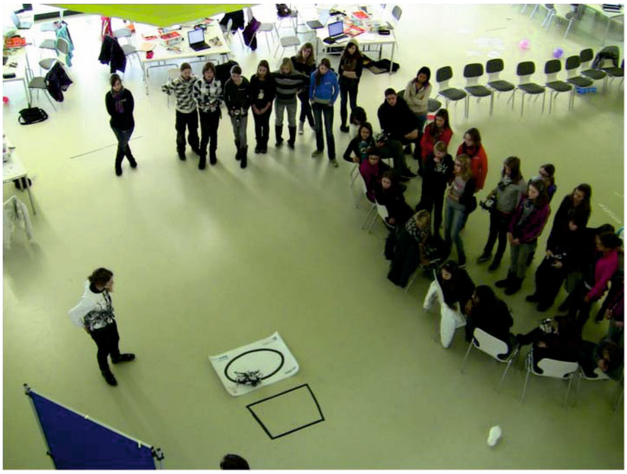

Fig. 1. Workshop impression. The robot is programmed to follow the bla

physical attributes of single components all the way to fully integrate
tions of industrial machinery. Simulating what is essentially a toy, and
physical accuracy, this simulator falls on the low end of this scale.

3.1 Robot Simulators

Simulators for Lego Mindstorms already exist, although they have diffe
and limitations as the one developed here. RobertaSim [15] is a sim
the RCX microcontroller (the deprecated LEGO Mindstorms RCX set
crosoft Windows platforms. It was developed for courses specifically
teaching programming to girls. The goal is to have a robot move thro
tual environment executing the same code that works on real robots. T
it similar to the go4IT!-simulator developed in our project. However,
work as a replacement, as it simulates a different hardware generat
thermore, RobertaSim focuses on physically accurate simulation, whi
it more complex to use and thus less suitable for our computer scie
ing scenarios. In our teaching scenarios the focus lies on the underst
algorithms (e.g. path finding algorithms) while the increased accurac
a simulation usually leads to different kinds of problems that must
(e.g. a robots sensor can be caught by a wall tile). Additionally it lack
support, thus it cant be used in collaborative learning scenarios.

"LMS" (Lego Mindstorms simulator) simulates Mindstorms NXT c
[13]. It requires the programs to be written for a custom firmware.
ulation aims at complex scenarios and offers a fairly extensive user
While sharing many of our goals on the technical side, this simulator i
complicated than ours and was not designed for teaching school child
also requires a custom firmware to be installed on the robots which
compatibility with the development tools we use in our courses.

At the higher end, simulation of robots can become extremely pr
complex. An example for this is SimRobot [14]. Here, complex robo
defined including sensors, various joints and even simulated cameras.
add more elements by implementing their own modules in *C++*. The
a wide variety of scenarios that can be modeled through this extensibi
simulator is clearly more advanced than the one described here, but
plexity also makes it unsuitable for the basic teaching task that the si
meant to fulfill in our scenario.

3.2 Other Microworld Approaches

The Java Hamster programming model or the Greenfoot microworld [1,
similar approach in teaching the basics of programming to children. H
program a cybernetic beastie such as a virtual turtle on the comput
using a subset of the Java language. The feature set of the hamster

our simulator the motors have to be turned at different speeds to execu
while the hamster simulator has a direct turn instruction. This is, in
adoption of Seymour Paperts microworld approach [10] for teaching
science to school children.

In our course concept we intentionally use a programming model tha
the users to program individual sensors and actuators in a C-like synta
hiding the actual programming complexity by providing abstractions
pre-defined commands like "`turnLeft()`" or puzzle programming meta
in Scratch [11]. We found that the participants of our courses enjoy prese
more difficult looking C-like code to their classmates and their parents
this leads to an increased self-concept in dealing with and interest in te

A number of approaches have been made to simulate robots in ge
Lego Mindstorms in particular, but none fulfill our requirements for sim
ing scenarios as described in the following.

4 Requirement Analysis

Our main goal was the development of a simulator for LEGO Mindsto
robots that can easily be used by all participants of our robotic cour
led to a number of requirements from a technical and a usability pers

4.1 Hardware Requirements

First, we strived for a platform independent solution that operates on
Windows, Mac OS X and Linux systems. An important goal of our
project is to have an appealing 3D output. To guarantee cross-platform
ibility we decided to implement all of this using OpenGL.

The participants of our courses come from different schools and bac
Therefore, we cant assume modern computing machinery to be availa
school students. We expected that many of the target systems will
tively little processing power, so we decided to use as little OpenGL fe
possible while still maintaining a clean code base. To ensure this, only
included in the 1.5 version of the OpenGL specification were used. T
features such as vertex buffer objects for management of geometry
does not yet include any form of pixel- and vertex shaders that may ca
lems on older hardware, for example with integrated GPUs. Furtherm
features included in OpenGL ES 1.1, a subset of OpenGL 1.5 specifica
at embedded systems, were used. In the future this might allow us t
new teaching scenarios that incorporate simulation based learning an
devices.

robots in the courses. Consequently, it should read the same bytecod produced by the development environment. It should also be possible launch the simulator directly from the development environment that our courses.

The simulator should be easy to deploy. Consequently the simulator be executable without the need for an installation routine. No setting stored in the Windows Registry or other local configuration files. The ap should not need extra privileges that would require an administrato account.

After the end of the course the simulator and the other developm should be handed out to the participants. They must be able to easil development and simulation environment when at home.

The number of user interface elements should be kept minimal and th require little or no explanation. To make the user interface that simpl tains only five buttons by default. The only more complex part is a s lection view that can be opened to configure the robot (see Figure 3). no modality: While the configuration view is opened, all other functio user interface remain fully usable and the program is not paused. Ap the window title required by the operating system, the user interface contain any text at all. Thus the program can be used regardless of l the user may speak and without needing any localization for the conte

4.3 Simulation Requirements

As we focus on teaching computer science concepts, it is not necessary late many different robot models or even allow full Lego construction. as different sensor placements obviously have a profound impact on rithms used to solve a given problem (e.g. path finding algorithms), it s possible to change the sensor placements easily. Likewise, the virtual ronment should be easy to edit, so that the algorithms can be tested in environments (like different maze setups). As the focus should be on s programs, not physics, physical simulations can and should have low a

The simulator is able to emulate a reasonably large subset of all pro the NXT system. Nevertheless some rarely used commands of the NXT need to be implemented. Of course all language features taught in ou courses must work within the simulator.

4.4 Additional Features

Our robotic courses are group learning experiences in which two stude a laptop and a robot and work together and different teams and the also interact with each other in common tasks. This collaborative grou experience should also be possible for students using the robot simula

on the local network. If one is found, the simulator automatically co
this instance and the environment is shared. If no other simulator is fou
local network, the client automatically becomes a server and starts to
itself to future clients. The network layer is targeted at local environme
auto discovery usually works. There is currently no user interface to
configure network settings to allow connections to arbitrary IP-addre
connections over the Internet), even though that would be technically

5 Simulator

The go4IT!-simulator is an interactive program with a GUI and 3D
simulated robots. It is written entirely in *C++* to support as many
as possible. It uses OpenGL for drawing, OpenAL for sound out- and i
SDL for the interaction with the operating system. At the core of th
an interpreter executes the compiled Lego NXT bytecode.

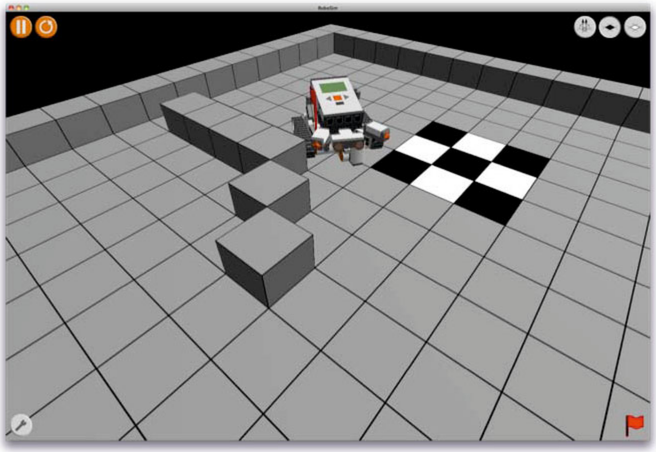

Fig. 2. Screenshot of the simulator with a single robot, a wall and shaded

5.1 Environment

The environment of the simulated robot is designed to be primarily
edit, allowing to create a wide range of scenarios in little time. In our
the environment is a grid of 20×20 cells by default. Grid-based approa

robot can drive on. Touch and ultrasound sensors will detect contact or
to the walls. To allow the light sensor to work with meaningful inputs,
has an arbitrary shade of grey (rendered in the user interface as black
white).

All the attributes of a cell can be changed at any time by clicki
Through the user interface, one can change between height mode (a cli
whether the cell is wall or floor), lighten mode and darken mode (
upper right corner). This makes it easy to create mazes for the robot to
through or to create colored lines that the robot should follow. Both
typical for our robotic courses.

The editor does not allow the cells at the edges of the map to be
to floor cells, although their colors can be altered. This prevents the ro
steering into the void.

5.2 Robot

There is only a single robot model available, using two of the three moto
for propulsion (a model with two wheels is used). The robot is simulat
suming essentially infinite acceleration and ignoring masses, which is a s
imprecise approach. However, the mass of a real robot is low enough c
to the total power output of the robot that this proves to be an ade
proximation. This is also aided by the fact that the speed of the moto
real robot is electronically regulated to reach a given speed as soon as
and then keep it approximately constant. As a result, the difference t
realistic simulation is small.

The robot can have up to four sensors registering touch events, ligh
noise levels or distances to an obstacle. Any combination of these s
possible, and the sensors can be freely placed on a circle around the
sensor may point forward or downwards. Its configuration can be ch
any time by opening the sensor configuration panel (see Figure 3). Ope
panel does not pause the execution of the program. Anything that can
while the configuration panel is not shown can also be done when it
(e.g. altering the environment by adding a wall).

The execution of a robot program can be paused, which also stops t
It is also possible to reload the program of the robot, which also res
execution from the beginning. A robot can be picked up and placed so
else via drag and drop. This feature is useful in case the robot is stuck
maze and the students want to check if the algorithm is working cor
a different position on the map. In case of a networked scenario, all
applies to the robot controlled by this particular computer. To sign
computer owns which robot, each robot has a uniquely colored flag.
the same color is displayed in the user interface of the computer contro
robot (see Figure 2, lower right corner).

Fig. 3. Sensor configuration panel. Sensor 1 is configured as ultra-sound senso
sideways. Sensor 2 is configured as a touch sensor pointing ahead. Sensors 3
disabled.

5.3 Sensor Values

Generating accurate sensor values is highly important, as this is the or
of input available to programs running inside of the simulator. The
offers the four types of sensors that are part of the LEGO Mindstor
sets: Touch sensors, light sensors, sound sensors, and ultra-sound sens

Typical programs running on a LEGO Mindstorms NXT robot us
to read sensor values. For example, a program that lets the robot follo
line is usually implemented as follows: At first it will turn on the mot
it will constantly check the light sensor until a certain threshold is
(i.e. `until(SensorLight(IN_3) > 40);`). This would trigger several
recalculations of the virtual sensor values per frame of the simulation
Hence the simulator calculates the value of each sensor once on every
of the run-loop, regardless of whether and how often it is actually reque
caches it until the next iteration of the run loop. As environment and
only change once per run through the run-loop at most, this deliver
results without calculating sensor data numerous times.

Each sensor is mounted to the robot at a specific angle, has a certa
and may point forward or downwards. This combination creates a speci
position, based on which the results are read. All sensors work the same
if they point forward or downwards, only with different directions. Ho
with real LEGO Mindstorms NXT robots it hardly makes sense to set t
ultrasound, or sound sensor to point downwards.

Ultrasound Sensor. To find the value for the ultrasound sensor, a r
from the sensors position. It is checked against the environment and oth
and shortest distance to any object hit by this ray is used as the sens

Touch Sensor. For the touch sensor we used collision detection metho
on the sensor position, a bounding box for the active part of the sens
erated and tested against the environment and other robots. The rob
get moved if such a collision occurs, but the sensor value is set correct

Light Sensor. The light sensor also casts a ray through the environ
reports the shade of the cell hit, or 0, for black, if no cell is hit. For ca
the light sensor value, any robot in the way is ignored, as it is generally
to find a color value for them. In our go4IT!-tasks, light sensors are
use pointing downward to read color information on the floor.

Sound Sensor. The sound sensor does not directly interact with the
ment. Instead, the simulator calculates a value, based on the positi
sound sensor, the positions and volumes of all robots playing sounds
noise reported in from an attached audio source like the computer mi
This allows the users of the simulator to build applications that let the
robot react on real-world events like clapping in the hands in front of
puter. The sound sensor also reacts to sounds emitted by other robots
in network mode (see below).

5.4 Network Mode

It is possible to run the simulator alone, with a single robot, or toge
others over the network, so that every robot gets shown on every compu
computer in the network simulation executes its own code and tran
state changes to the server. Conceptually, there is a difference between
server, which calculates the physical simulation for all robots, and cli
just execute code, transmit its output and then display robots at the
given by the server, but from the users view, both work the same.

5.5 Collision Detection

As robots cant pass through walls or other robots, the simulator uses
detection. For this an oriented bounding box of the robots is calculat
bounding boxes are also used to provide input values for the touch s
the ultra-sound sensor. In that case a ray is cast from the sensors. In
of the ultra-sound sensor the reported distance is the distance from th
point of the ray to the nearest point where a bounding box intersects
casted ray. As the robot model is not a perfect box, this can result in
in which a collision is reported even though the ray might pass the rob
but not the bounding box. In practice this is not a problem as the a
robot is comparably large.

two robots collide, both robots are moved so that they no longer collid
done without changing the driving direction.

All this happens in 2D only, as the environment design does not a
robot to leave the ground floor. In the special case of a robot being pic
move it somewhere else, all collision detection is disabled.

Using this method, no momentum, energy, elastic or plastic prop
regarded. If a robot hits a wall, it simply stops moving any further, and
the wall at an angle, it will glide along the wall.

It is possible to implement a more accurate collision model for the s
however, a more accurate representation of physics would require use
only develop programs, but also keep in mind the physical properti
robot. This higher grade of complexity, however, would increase the ex
load of the learners and would hinder their ability to focus on the dev
and implementation of algorithms [3].

5.6 Integration into the Development Environment and Pro

The simulator described above works well as a stand-alone simulation e
Lego Mindstorms NXT bytecode. However we wanted to closely inte
simulator into the development process the school students learned
during the courses. There the students usually apply an iterative dev
cycle in which they start off with defining a goal (e.g. let the robot
dark line). Then they implement the corresponding program and test
program does not work as expected – which is usually the case –, they
to the implementation phase. In our courses this iterative developmen
has rather quick cycles, as the deployment of an adjusted program on
can be done within seconds over a Bluetooth connection.

To allow equally fast development cycles when the simulator is used, w
integrated the simulator in the BricxCC[2] integrated development env
we are using in the courses. This modification allows the students to a
cally deploy a program on the simulator once it is compiled.

6 Evaluation

There were two main parts of the testing. First of all, the simulator w
on a number of different computers and different operating systems
that it is platform independent and easy to deploy. The simulator w
with a number of robot programs that the school children developed d
courses. The programs ranged from rather simple programs that let t
drive simple geometric shapes without reacting to sensory input to n
maze solvers like Pledges Algorithm.

[2] http://brixcc.sourceforge.net/

Up to now, the simulator has not been used in our actual go4IT!-w
and just put on the web site for download. Thus, there have not
evaluations of its utilization as a follow-up learning tool.

6.1 Technical Evaluation

To evaluate if the simulator works platform independent, it was tes
different computers and using different NXC programs. It was tested su
on multiple Mac OS X versions (10.5 and 10.6) using either a PowerP
an Intel CPU. It was also tested with Windows XP and Windows 7 ru
an Intel CPU. It worked well with GPUs by ATI and NVIDIA, as we
an emulated GPU used in a virtual machine.

A number of programs from the robot workshops were gathered. Th
ations in the simulator were compared to a real Lego Mindstorms NX
Indeed the programs running on simulated robots worked similar to
running on real robots.

Finally, testing was also done with network support between all testi
systems, as well as locally between two instances running on the same

On all platforms, the simulator was able to execute all test progra
rately and produce results corresponding closely to the ones observed
real robot. Network sessions also worked between any of the systems,
of the systems working either as server or client. Networking was al
with multiple clients and worked as expected there.

No precise performance measurements were gathered, but the simulat
without any noticeable slowdowns on all target systems, including in
mode. The graphics results were generally identical and comparable t
servations in the real world.

6.2 User Testing

Finally we invited a group of school students of grade ten to twelve t
simulator. In addition to the normal BricxCC IDE and a LEGO Mi
NXT robot, the simulator was offered to the students. They were ask
any problems they noticed.

The students confirmed that such a simulator is a very useful aid b
allowed faster write-test cycles. They also agreed that the simulator is
repeat the subject matter from the workshops at home.

The students also discovered a number of smaller issues and a few
flaws. In particular they criticized:

– It is not possible to remap the motor ports, so code written for a r
using different ports for the motors does not work directly.

– Some more advanced operations of the NQC programming langu
not implemented.

These issues have been fixed shortly after the user test.

7 Summary and Future Work

The developed simulator is able to execute a wide variety of NXC
and allows creating environments for simulated Lego robots to test t
grams. Following the requirements, more complex programs cannot be s
accurately, although it should be possible to extend the amount of supp
erations in the future. Networking works as expected, and there is no f
difference between any of the supported platforms. With that, the
fulfills all requirements as a teaching tool.

The simulator is developed to be a teaching tool, and as such the s
need not to be exceedingly precise. Also it is not necessary that the
bytecode and all advanced features are supported. For future work,
scope could be expanded to make it a fully featured simulator for all Le
storms robots and programs. This could allow both teaching more
programming techniques and using it to simulate other projects fast
so, first of all, current limitations that were acceptable within the sco
project would need to be removed. For example, it should be possibl
port a larger portion of the byte code. In addition, the current simple
simulation could be replaced by a more robust one, possibly using libr
as Bullet, adding realism, but making programming harder as users w
to account for physical reactions. In such a situation, it would also be
to have more robots to choose from.

The environment systems could be extended in various ways. Othe
for the robot to interact with would allow for new, more complex scen
are currently impossible. The same would apply to an environment w
actions possible, such as ascending to a higher point by means of ra
would necessarily make it more complex to modify the environment an
require a completely different approach to editing.

For such a tool, more efficient networking and Internet support m
be desirable. This would require making the network protocol more e
replacing it completely, and adding measures against high transmiss
and improved synchronization techniques.

Since all the libraries used in the simulator are available on othe
ing systems, it should be possible to port the simulator to system
GNU/Linux rather easily if desired.

(2008)

2. Brauner, P., Leonhardt, T., Ziefle, M., Schroeder, U.: The effect of ta tifacts, gender and subjective technical competence on teaching progra seventh graders. In: Hromkovič, J., Královič, R., Vahrenhold, J. (eds.) IS LNCS, vol. 5941, pp. 61–71. Springer, Heidelberg (2010)
3. Chandler, P., Sweller, J.: Cognitive Load Theory and the Format of In Cognition and Instruction 8(4), 293–332 (1991)
4. De Boer, W.H.: Fast Terrain Rendering Using Geometrical Mipmapping `http://www.flipcode.com/tutorials/tut_geomipmaps.shtml`
5. Eggert, D.W.: Using the Lego mindstorms NXT robot kit in an introdu programming class. J. Comput. Small Coll. 24(6), 8–10 (2009)
6. Hartmann, S., Schecker, H.: Bietet Robotik Mädchen einen Zugang zu In Technik und Naturwissenschaft? – Evaluationsergebnisse zu dem Projekt Zeitschrift für Didaktik der Naturwissenschaften 11, 7–19 (2005)
7. Kölling, M., Henriksen, P.: Game programming in introductory courses v state manipulation. In: Proceedings of the 10th annual SIGCSE conf Innovation and technology in computer science education. ITiCSE 200 63. ACM Press, New York (2005)
8. Leonhardt, T., Brauner, P., Siebert, J., Schroeder, U.: Übertragbarkeit MINT-Interesse-initiierender außerschulischer Maßnahmen. In: INFOS GI-Fachtagung Informatik und Schule (September 2011) (accepted)
9. Leonhardt, T., Schroeder, U.: go4IT!: Initiierung und nachhaltige Förd Interesse an MINT-Fächern bei Mädchen. In: Informatische Bildung i und Praxis, Beiträge zur INFOS 2009, 13. GI-Fachtagung - Informatik u Berlin (2009)
10. Papert, S.: Mindstorms: Children, Computers, and powerful Ideas. Ba Inc., New York (1980)
11. Resnick, M., Maloney, J., Kafai, Y., Rusk, N., Eastmond, E., Brennan, K A., Rosenbaum, E., Silver, J., Silverman, B., et al.: Scratch: programmi Communications of the ACM 52(11), 60–67 (2009)
12. Schinzel, B.: Informatik und Geschlechtergerechtigkeit in De Annäherungen. Reihe Gender Studies (Gender and Science, Perspe den Natur- und Ingenieurwissenschaften), 127–146 (2007)
13. Schmalzbauer, J., Scheel, O.: Lego Mindstorms Simulator (2011)
14. Siems, U., Herwig, C., Röfer, T.: SimRobot, ein System zur Simula sorbestückter Agenten in einer dreidimensionalen Umwelt. No. 1/94 Bericht, Zentrum für Kognitionswissenschaften. Universität Bremen (19
15. Theidig, G., Börding, J., Petersen, U.: Roberta - Der Simulator RobertaSi hofer IRB Verlag, Stuttgart (2006)