

# A Simulator for Teaching Robotics Programming Using the iRobot Create

Andrew Hettlinger, Matthew R. Boutell

Department of Computer Science and Software Engineering  
Rose-Hulman Institute of Technology  
5500 Wabash Ave., Terre Haute, IN 47803  
hettliaj@gmail.com, boutell@rose-hulman.edu

## Abstract

Past educational robotics research has indicated that the use of simulators can increase students' performance in introductory robotics programming courses. In this paper, we introduce a simulator for the iRobot Create that works on Windows PCs. It was developed to work with a Python robotics library and includes an Eclipse plugin, but can simulate any library that uses the serial Open Interface on the Create. The platform, library, and simulator are all easy to use and have been well-received initially by students.

## Motivation

The use of robots in education continues to rise. A growing number of undergraduate institutions are incorporating robotics into their first programming courses as a means of attracting students. While robots do draw students, their hardware adds complexity: they can break and are not as portable as a computer alone. Fagin and Merkle (2002) argued that the lack of 24/7 access to robots hindered student learning because students did not have sufficient time to practice the write-run-debug feedback loop. They hypothesized that a simulator would help with this, and a later paper indicates this was the case (Fagin 2003).

While exceptions exist (Summet, *et al.* 2009), robots often must stay in a lab and be shared by students. A simulator can offer a level of convenience similar to that enjoyed by students in other programming courses; that is, they can do most of their programming *wherever* they like *whenever* they like; say, in their dorm room late at night. And while the ultimate goal for many students is to program a physical device, a simulator allows programming concepts to be decoupled from the hardware during early phases of development.

We created a robotics version of the introductory software development course at our institution in Fall, 2008, as part of a new Robotics minor. We wanted to give students an exposure to robotics as early as possible in the curriculum, so needed a platform and library that were easy to learn and to use, both for the students and the instructor. For a platform, we chose the iRobot Create. At \$280 per

robot including a Bluetooth module, it is much cheaper than research-grade robots, yet is commercial quality. It can be programmed by transmitting serial commands using iRobot's Open Interface, and libraries have been developed for various languages to interface with it (Blank, *et al.* 2004; Dodds 2007; Esposito, *et al.* 2008). Our first computing course uses Python. Pyrobot has nice simulators (Stage, Gazebo, and Pyrobot) and is very versatile, being cross-platform and with modules for robots of all types, including the Create. However, we ultimately chose to use the pyCreate library (Dodds 2007), which works only on the Create, since we valued simplicity over flexibility.

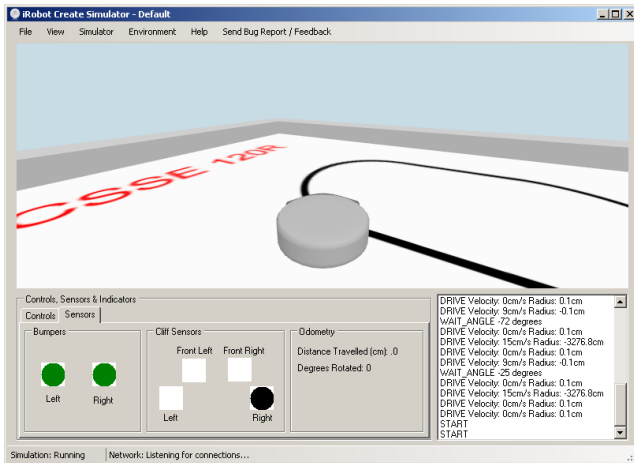
## Simulator Details

A goal of the simulator is to provide a reasonably-realistic visual simulation while still operating in real-time. While a 2D simulation would have sufficed for these purposes, we chose to implement it in 3D using Microsoft's Managed DirectX API. This allowed us to take advantage of fast hardware rendering and gave us the ability to implement several viewpoints rather than one flat view. We created four different views of the virtual robot in its environment. *Follow*, with the camera slightly behind and above the robot (Figure 1), and *onboard*, with the camera "riding" on top of the robot, are similar to those employed in many video games. *Top-down* uses an orthographic projection. *User* is customizable, using the mouse to position the camera at any desired location.

The environment surrounding the robot is simple but flexible, consisting of a rectangle with a texture that serves as the floor and any number of cylinders or rectangular prisms which serve as obstacles. The location, orientation, size, and color of each object along with the background image are defined in xml and can be customized using an environment editor.

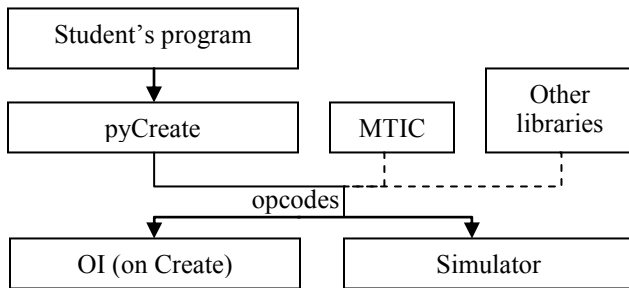
The simulator models many of the Create's sensors: the four cliff sensors and the two bump sensors are shown in Figure 1, with digital values shown graphically in the lower-left portion of the screen. Sensors for the play and

advance buttons are also implemented. The buttons and their corresponding LED values appear in a different tab.



**Figure 1:** The simulator interface.

Our students' code is developed in Python using high-level commands such as `robot.go(degrees)` in the pyCreate library. PyCreate then translates these commands into a series of opcodes defined in the Open Interface (OI) and transmits them to the Create (Figure 2). We made a small modification to pyCreate to send a duplicate stream of opcodes to a network socket on which the simulator is listening. The benefits of using such an approach are several: communication is asynchronous, a real robot and the simulator can be controlled simultaneously, and any external system can be used to drive the simulator. The last benefit may be the most important. Any system that would like to control the simulator simply needs to open a socket on the correct port and send valid opcodes. It is thus Python-independent, and requires only small modifications to work other libraries such as MTIC, the MATLAB toolbox for the Create developed by Esposito, *et al.* (2008).



**Figure 2:** System architecture. Planned updates are shown as dotted lines.

Due to its implementation in C# using DirectX, the simulator only runs on Windows-based PCs. However, the resulting straightforward installation fits our goal of simplicity for beginners.

The simulator does a reasonable but inexact job of modeling the real world. The environment's lighting is fixed, so thresholds must be changed for code developed in simulation to work with real-world lighting. It does not

have a sophisticated physics engine to account for wheel slippage or friction. The sensors aren't exact replicas of the real robot's. However, our goal was never to replace the robot altogether. We don't want to discourage students from using the robot to test! But we do want to provide them with an alternative during early development. So while they might need to tune thresholds using the robot, they can perform the time consuming and sometimes-frustrating aspect of development for beginners (dealing with logic errors) at their convenience using the simulator.

## Discussion and Conclusion

The simulator was introduced this Fall and was received well by students. One student in particular who had trouble connecting to the robot with his Bluetooth module commented that he loved using the simulator because he was still able to continue developing instead of relying on his partner's laptop. The simulator is also currently being used in a course at Cornell University.

Simulators can provide a level of convenience to students, decoupling the physical robot from the early stages of development. We have developed a 3D, Windows-based simulator for the Create robot. We chose this platform and library due to its simplicity, and developed the simulator with the same goal in mind. We conjecture that other educators with this same goal would likewise benefit from this simulator. The simulator is available at <http://www.rose-hulman.edu/class/csse/resources/Robotics/InstallingCreateSimulator.htm>.

Future work includes an evaluation of students' use of the simulator and extending it to libraries in other languages such as MTIC.

## References

- Blank, D.S.; Kumar, D.; Meeden, L. & Yanco, H. 2004. Pyro: A Python-based Versatile Programming Environment for Teaching Robotics. *J. Educ. Resour. Comput.*, 4(3).
- Dodds, Z. pyCreate. 2007. [www.cs.hmc.edu/~dodds/erdos/old.html](http://www.cs.hmc.edu/~dodds/erdos/old.html). Copyright 2007.
- Esposito, J. M., Barton, O.; and Koehler J. 2008. Matlab Toolbox for the Create Robot. [www.usna.edu/Users/weapsys/esposito/roomba.matlab/](http://www.usna.edu/Users/weapsys/esposito/roomba.matlab/). Copyright 2008.
- B. Fagin. 2003. Ada/mindstorms 3.0: A computational environment for introductory robotics and programming. *IEEE Robotics and Automation Magazine*, 10(2):19-24.
- Fagin, B. S. and Merkle, L. 2002. Quantitative analysis of the effects of robots on introductory Computer Science education. *J. Educ. Resour. Comput.* 2(4).
- Summet, J; Kumar, D; O'Hara, K; Walker, D; Ni, L; Blank, D; Balch, T. 2009. Personalizing CS1 with Robots. *Proc. ACM SIGCSE*. Chattanooga, TN USA.