# DOROTHY: Integrating Graphical Programming with Robotics to Stimulate Interest in Computing Careers

David South, Austin Ray, Kevin Thomas, Stephanie Graham, Shiloh Huff, Sarah Rainge
Texas Tech University
Department of Computer Science
Lubbock, TX 79409
936-827-4876
david.south@ttu.edu, austin.ray@ttu.edu kevin.thomas@ttu.edu stephanie.graham@ttu.edu shiloh.huff@ttu.edu, sarah.rainge@ttu.edu

Mary Shuman
University of North Carolina, Charlotte
Department of Computer Science
Charlotte, NC 28223
704-345-8034
mshuman@uncc.edu

Sabyne Peeler
Florida A&M University
Department of Computer and Information Sciences
Tallahassee, Florida 32307
850-599-3042
speeler4@yahoo.com

Mohan Sridharan[1]
Susan D. Urban[2]
Joseph E. Urban[2]
Texas Tech University
[1]Department of Computer Science
[2]Department of Industrial Engineering
Lubbock, TX 79409
806-742-2484
mohan.sridharan@ttu.edu
susan.urban@ttu.edu
joseph.urban@ttu.edu

## ABSTRACT

This paper describes DOROTHY, a novel educational tool that enhances the Alice 3D programming environment to enable bidirectional communication of sensor data and commands with robots capable of autonomous operation. Users without any programming experience can quickly create graphical routines consisting of one or more simulated robots in virtual worlds. Command dictionaries and socket streams enable real-time translation of these routines to software for synchronous or asynchronous control of sensing and actuation on one or more mobile robots with on-board sensing, resulting in adaptive behavior in the real-world. Multiple execution scenarios are described to illustrate the capabilities of the educational tool. Furthermore, the paper outlines a curriculum that can be used with the tool to teach core concepts of computing, concurrent execution and real-world sensing to middle school and high school students, thus stimulating interest in computing.

## Categories and Subject Descriptors

D.2.6 [**Programming Environments**]: Graphical environments, integrated environments; I.2.9 [**Robotics**].

## General Terms

Design, Documentation, Experimentation.

## Keywords

Computational thinking, graphical programming environments, autonomous mobile robots.

## 1. INTRODUCTION

The technological and economic growth of a country depends on training computing professionals drawn from the full spectrum of diverse groups that compose the country [2, 3, 16]. However, many communities face long-standing under-representation in undergraduate and graduate programs in computing disciplines. These programs also have low retention rates, especially for students from under-represented groups. Research indicates that many students have made career choices by the time they are in high-school [4]. Middle school and high school students therefore need to be engaged in projects where they actively learn and apply computational thinking concepts [23]. Enabling these students to discover the satisfaction of successfully addressing real-world computing challenges may encourage them to pursue advanced degrees and careers in computing.

A plethora of programs have introduced K-12 students to computing concepts in an effort to stimulate interest in computing. Programming courses continue to serve as a significant means of introducing students to computing, but many students have been overwhelmed with the syntax of programming languages. Students also lose interest in computing because many introductory courses fail to connect computing with real-world applications. In recent years, 3D graphical programming environments such as Alice [1], Scratch [19] and Greenfoot [8] have been a more effective way of teaching students to program. Middle school students have also been introduced to *storytelling* as a way to learn programming concepts in 3D environments [12]. Stories consist of action sequences (i.e., animations) generated in simple pseudo-code based on virtual characters, objects, and scenarios. However, these environments do not support real-world enactment of the graphical routines, and students using these environments may fail to make the transition to advanced computing concepts and their application to real-world problems.

Many students have also embraced the world of robotics as an introduction to computing [11, 13, 14, 18, 20]. Robots are great for illustrating practical applications of core computing concepts. However, some students find it challenging to work with physical robots and struggle to learn the complex syntax required to program robots. In addition, many existing robot kits limit the students' ability to obtain a deeper understanding of computing that can result by programming robots capable of autonomous operation using sensor feedback.

The Alice 3D environment has been used as a programming interface with mobile robots to successfully stimulate interest in CS1 courses within an undergraduate computer science program [6, 21, 22]. However, existing schemes typically interface with a single robot, do not support bidirectional communication between Alice and robots, or do not support graphical routines that result in autonomous and adaptive behavior on robots. This paper presents DOROTHY (*Design of Robot Oriented Thinking to Help Youth*), an educational tool that addresses these challenges by customizing and enhancing Alice, making the following novel contributions:

1. A command dictionary and a socket stream enable real-time translation of graphical routines (i.e., action sequences) in virtual worlds to corresponding sensing and actuation commands on one or more wheeled robots capable of autonomous operation using on-board sensor inputs (e.g., camera and infrared sensor).
2. A robot handler provides bidirectional communication of sensor data and commands between robots and Alice, enabling synchronous (i.e., concurrent) or asynchronous execution of graphical routines in virtual worlds and on physical robots.
3. A modular architecture allows socket communicators to be written in any programming language. The tool can be used with different types of robots by changing the command dictionary.

DOROTHY thus fully exploits the complementary features of robots and graphical programming environments. Students program in virtual worlds using syntax that is easy to learn and see the physical enactment of the program on robots in the real-world. Furthermore, we have developed a curriculum for using the tool to teach core concepts of computing, concurrent execution and real-world sensing to middle school and high school students.

The remainder of this paper is organized as follows. Section 2 presents related work on using Alice and robots for computing education. Section 3 describes DOROTHY's modular architecture that enables real-time bidirectional communication between virtual domains and physical robots. Section 4 presents illustrative scenarios and outlines the curriculum that can be used to teach core computing concepts. Section 5 presents a summary of initial efforts and future plans of using the tool and curriculum to teach computational thinking to middle school and high school students.

## 2. RELATED WORK

This section discusses previous work that has motivated the development of DOROTHY. Section 2.1 provides an overview of the Alice programming environment; Section 2.2 describes the use of robot platforms for computing education; and Section 2.3 describes previous research on integrating Alice with robots.

### 2.1 3D Programming with Alice

Graphical (i.e., 3D) programming tools are becoming increasingly popular for teaching programming concepts due to their easy-to-use interfaces and simple programming syntax [1, 6, 8, 19]. Users create visual representations of virtual worlds by adding virtual objects to the scene. These objects are manipulated by action sequences (i.e., graphical routines or programs) created by drag and drop techniques or pseudo code. Students learn and use computational thinking to achieve the desired functionality.

Our tool builds on the Alice 3D programming environment. Alice is a system developed by researchers at Carnegie Mellon University [1, 5] that uses storytelling to maintain user interest while teaching introductory programming concepts. Alice provides multiple virtual objects in a large pre-existing gallery, along with the ability to create new 3D virtual objects. A user can create a story (i.e., an action sequence) with these objects using pseudo Java code, which illustrates data structures, functions, control structures and the correlation between stories and programming concepts. In congruence with the idea of storytelling, Alice programs are called 'worlds'.

### 2.2 Robots for Computing Education

Robots are great tools for stimulating interest in computing since they illustrate real-world applications of computing concepts [13, 14, 20]. For instance, the *Advancing Robotics Technology for Societal Impact* (ARTSI) alliance led by Spelman College focuses on increasing the number of African American students in computer science. The Lamar University *INcreasing Student Participation in Research Development* (INSPIRED) program uses Scratch and robots to motivate under-represented middle school students to pursue computing education [7]. The *Hispanic Computer Brigade* at San Jose State University uses Alice and pico-crickets to inspire Hispanic students to pursue computing [9].

The primary robot platforms used in this project are the *fluke* robots created for the Myro (My Robotics) project by the Institute for Personal Robots in Education [10], a joint venture between Georgia Institute of Technology, Bryn Mawr College, and Microsoft Research for promoting the use of robots in computer science education [11]. These robots consist of an add-on board mounted on a scribbler wheeled robot base with a microcontroller [15]. The wheeled base can move forward, backward, and turn. The base can also be fitted with a marker to draw patterns. The add-on board provides inputs from sensors, such as a camera, infrared sensors, and light sensors, and enables communication between the robot and a computer using Bluetooth technology. The Myro project provides a framework with many built-in functions that can be used for I/O, movement, obstacle detection, sensor data acquisition, and image processing on fluke robots. Although versions of the Myro framework are being developed in different programming languages, DOROTHY interfaces with the Myro framework written in the Python programming language.

### 2.3 Integration of Robotics and Alice

Wellman, et al. [21, 22] integrated the Alice programming environment with the iRobot Create platform to stimulate interest in CS1 courses in the undergraduate computer science program at the University of Alabama. Using source code from Alice 2.0, the Java programming language and the Bluetooth-enabled robot platform (based on the Roomba robot base), researchers generalized the robot's commands to terms in Alice and created a component that utilizes Alice's output [6]. This program, called "Providing Robotic Experiences Through Object-Based Programming" or "PREOP", allows users to program an iRobot using Alice [17]. The iRobot, represented by a custom virtual version in Alice, can also be commanded to execute motion commands, play a song, or beep to a chosen frequency. Classroom modules were designed based on PREOP for an introductory computer science course. Studies show that the use of PREOP significantly increased the students' self-ratings of confidence and computing abilities [6, 22]. However, PREOP does not allow users to simulate sensing or create programs that result in adaptive behavior on the robot (e.g., automatically detect and track objects). Instead, the robot separately uses iRobot Create's bump detection to avoid obstacles. The PREOP program is also limited to operate on one instance of a specific type of robot.

Our educational tool significantly enhances Alice to allow bidirectional communication of data and commands with robots, translating graphical routines in virtual worlds to adaptive behavior on multiple (and different types of) robots. In addition, the tool enables synchronous (i.e., concurrent) or asynchronous (i.e., separate) execution of action sequences in virtual worlds and on robots in the real-world. Figure 1 shows some fluke robots, the re-designed DOROTHY interface and the *erratic* wheeled robot (with on-board sensors and processor) that has been partially integrated with DOROTHY. Users can thus design complex graphical routines and execute them on robots, learning advanced computing concepts beyond just basic control structures.

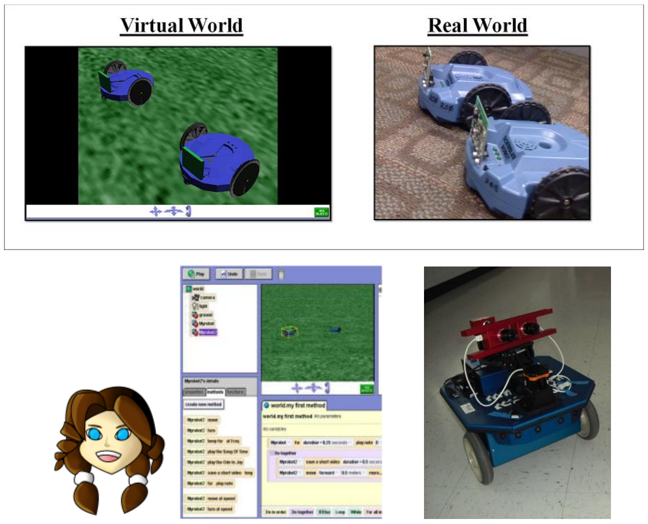


**Figure 1: Flukes and their virtual representation; DOROTHY interface screenshot; autonomous robot platform (*erratic*).**

## 3. DOROTHY ARCHITECTURE

This section describes the modular architecture used to create DOROTHY. Section 3.1 explains an initial XML-based approach that used Alice's ability to export its user-made programs into a web page. Section 3.2 discusses the revised real-time approach that uses a robot handler for communication between virtual worlds and robots, creating a more dynamic method for representing and manipulating robots in the virtual worlds.

### 3.1 XML Approach

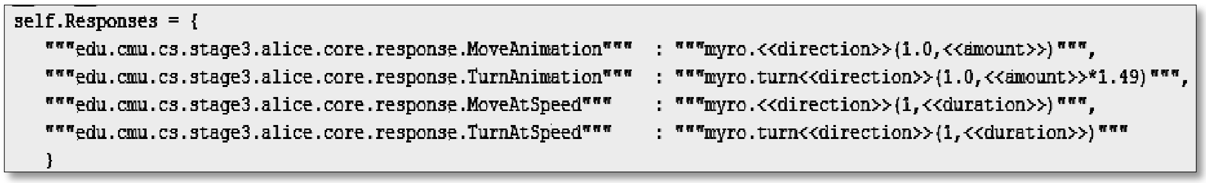
The initial approach investigated the development of a program to gather action information in Alice using the structure of Alice's world files. Alice saves its worlds in a zip file containing collections of XML files. Each XML file consists of a conditional statement, an event or a variable. The organization of world files is based on the nesting of blocks of code in an Alice world. Conditional statement directories contain folders of other commands or conditional statements, while command directories contain an XML file with information about that command. A Python program was designed to read saved Alice world files and parse XML tags from each page's source code. This information is translated into the Myro API and sent through a Bluetooth connection to the fluke robot. A dictionary data structure, as shown in Figure 2, supports this translation of an action sequence into the robot's programming language. Commands in the virtual world that are incompatible with the robot's commands are ignored. The translated commands are written to a Python script file that the robot can understand. Since the dictionary supports the ability to translate graphical routines to any robot language, multiple types of robots can be used in the virtual world and the corresponding physical world. The generated script file can be run on robots without installing Alice.

### 3.2 Robot Handler Approach

The XML parser approach does not support real-time editing or feedback from the robot. The user programs a virtual robot, generates the script file, and transmits the script file to the physical robot to see real-world enactment of the moves of the virtual robot. DOROTHY addresses this limitation by customizing Alice with a robot handler that uses sockets for bidirectional communication between the graphical environment and the robot. A user can program a virtual robot and have the physical robot move in synchronization. Multiple (and different types of) robots can be controlled concurrently using multiple handlers. The following subsections describe the robot handler approach.

#### *3.2.1 Intercepting Actions and Questions*

The robot handler was designed through the observation that objects (e.g., robots) in virtual worlds and the real-world can receive requests for information about their surroundings, and can be sent commands to perform actions. Execution of a routine in a virtual world creates events that tell objects to perform specific actions or answer questions. Figure 3 shows the data flow for actions and questions in DOROTHY. The corresponding events are distinguishable because they are handled in different locations. An action event is intercepted when the action is converted into a "Runtime Response" that uses all relevant information to act on a virtual object. The appropriate robot handler receives the information about the class/type of action, parameters needed for the action, and object/subject that will perform the action. The handler's *data receiver* passes this datagram to the *translator*. The translator initially reads the class-name of the action and compares it with a dictionary data type containing the class-names of all actions that the robot can perform. If a matching class-name is found in the dictionary, the corresponding action template contains a basic layout of the action and variables that need to be parsed from the datagram. The data extracted from the datagram is passed to the commands created for execution on the robot(s), e.g., for the flukes, commands are generated as function calls from the set of functions available for use in the Myro language.



```
self.Responses = {
    """edu.cmu.cs.stage3.alice.core.response.MoveAnimation"""  : """myro.<<direction>>(1.0,<<amount>>)""",
    """edu.cmu.cs.stage3.alice.core.response.TurnAnimation"""  : """myro.turn<<direction>>(1.0,<<amount>>*1.49)""",
    """edu.cmu.cs.stage3.alice.core.response.MoveAtSpeed"""    : """myro.<<direction>>(1,<<duration>>)""",
    """edu.cmu.cs.stage3.alice.core.response.TurnAtSpeed"""    : """myro.turn<<direction>>(1,<<duration>>)"""
    }
```

**Figure 2: Dictionary for DOROTHY-to-Myro translation.**

A question event is created when a user wants to check the condition of the virtual world, e.g., presence of obstacles in front of the robot. If the question can be answered by the robot as modeled by the robot handler, the question is intercepted and sent to the handler. The robot handler queries the robot's sensors for the corresponding data, e.g., infrared sensors to detect obstacles. This data is communicated to the graphical environment, effectively overriding the virtual object's observations and displaying real-world observations of robot(s) in virtual worlds.

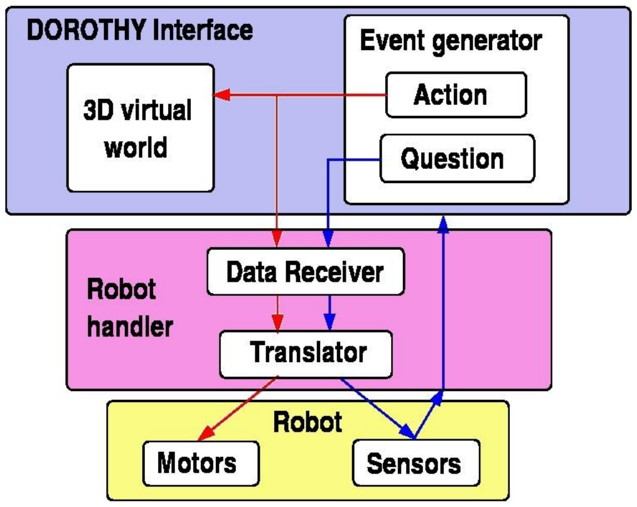


**Figure 3: Communication between graphical environment and Robot Handler in DOROTHY. Event generator and robot handlers enable bidirectional communication.**

### *3.2.2 Sockets for Bidirectional Communication*

The robot handler needs to listen for commands from the virtual world to the robot and communicate sensor information from the robot back to the virtual world. This capability is implemented using sockets, making it possible for the robot handler to be written in different programming languages. As the DOROTHY interface and the robot handler initialize, a socket stream is created to connect them. Only actions in a graphical routine (i.e., in the virtual world) that can be translated to appropriate actions on the robot are sent to the robot handler.

Events in virtual worlds are generated when a user executes an event script, e.g., when a user right clicks on a virtual object, they are able to command it to perform an action instantaneously. To provide a similar capability with the handler, the robot handler always listens for these events. As a result, the user can see the virtual robot(s) and real-world robots moving together.

The implementation of the robot handler ensures that definitions of robot capabilities can be edited and new robots can be added without making changes to the graphical environment. The modular architecture of DOROTHY also enables interaction with any robot capable of wireless communication (e.g., with Bluetooth technology). Adding a new robot just requires the addition of the corresponding functions (e.g., API options for a specific robot) to the translator's dictionary.

### *3.2.3 Using Multiple Robots*

In addition to supporting multiple robot types, DOROTHY also supports concurrent or asynchronous control of multiple instances of the same type of robot. This objective is achieved by adding a *robot tracker* file to the interface. When a new robot object is added in the virtual world, information about the object (e.g., name, type, and procedure to access this robot type) is stored in a *RobotType* object. The RobotType object for each robot is stored in a list. The object names to be saved to the list are determined automatically by checking if the corresponding object is of class *robot*, an extension of the standard object class in Alice.

Each handler runs in its own thread to connect each robot with its virtual representation. The initial setup is such that the handlers automatically establish connections with robots as and when they appear. The capabilities of the fluke robots currently integrated in DOROTHY include: (a) movement; (b) infrared sensing; (c) light sensing; (d) image processing and video recording; (e) computer speaker vocalization; and (f) real-time visualization of sensor readings from robots. A connection of up to five fluke robots has been tested with DOROTHY for synchronous and asynchronous operation. We have also enabled DOROTHY to send movement commands to a more sophisticated wheeled robot (*erratic*) using the Python Paramiko module and the SSH connection protocol.

## 4. EXECUTION SCENARIOS

Seven undergraduate students from three different universities developed DOROTHY. The students were supervised on individual research projects and summer internships since Summer 2011, with partial support from an NSF Research Experiences for Undergraduates (REU) site project. The objective is to develop an educational tool to stimulate interest in computing among middle school and high school students. This section describes some execution scenarios and outlines the curriculum that can be used with the tool to teach core computing concepts (e.g., abstraction, functions, control structures, iteration and concurrent execution) to school students. The scenarios are described below in the context of fluke robots.

### 4.1 Scenarios for a Single Robot

The basic scenario for the DOROTHY tool supports bidirectional communication between a virtual robot and its physical cohort. Graphical routines created in virtual worlds can have the robot move forward, backward, and turn. Such movements can be embedded in loops to teach sequential reasoning and iteration. The robot in the virtual (real) world can also sense objects in the virtual (real) world and adapt its behavior—this scenario can be used to teach control structures such as *if-then-else*. The virtual robot can also be programmed to respond based on the objects sensed by its cohort in the real-world. The available capabilities include: (1) developing an algorithm based on a virtual world; (2) developing a non-deterministic algorithm based only on the sensor inputs received by the robot in the real-world; and (3) developing an algorithm where the robot dynamically senses objects in the virtual and the real-world. The tool also supports the dynamic addition of objects in the virtual world as they are sensed in the real-world. Figure 4 shows a fluke executing an obstacle avoidance routine in a boxed-in area in the real-world. The robot uses infrared sensors to sense walls (i.e., obstacles) and responds according to instructions from the virtual world, e.g., the robot turns away from obstacles and determines a new direction of motion that is obstacle-free. Although not shown, a virtual robot executes the same movement in-sync with the physical robot.

Since the robot base can hold a pen and write on the floor, the DOROTHY interface can be used to teach the concept of a procedure, e.g., procedures for writing specific words or "patterns". More challenging scenarios for a single robot include: (a) searching for the brightest location in the room (using light sensors); and (b) locating objects of a specific shape or color (based on image processing), while navigating safely around obstacles. These scenarios require students to learn and use data structures and functions, in addition to algorithms for sensor input processing and navigation on robots.

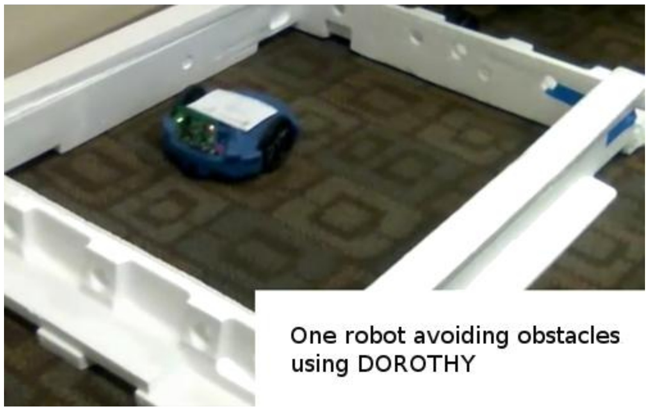


**Figure 4: Fluke robot detecting walls in a boxed area and communicating obstacle information to DOROTHY interface.**

## 4.2 Scenarios for Multiple Robots

DOROTHY allows for the creation of scenarios involving multiple robots that run simultaneously. The underlying Alice interface has a "Do Together" block that can be used to teach the concept of concurrent execution. Figure 5 illustrates three robots as they simultaneously move in a square pattern while avoiding obstacles. Such scenarios can be used to create challenging computational thinking exercises, where students can program multiple robots to work together to achieve a task, e.g., the synchronized motion of two or more robots to perform a "dance" step, and a team of robots on a "treasure hunt" or assembling a complex structure by retrieving component parts. Students will hence learn to use advanced data structures and functions, and sophisticated algorithms for sensor input processing and teamwork. An added benefit is that students working on such projects learn important teaming and social skills.

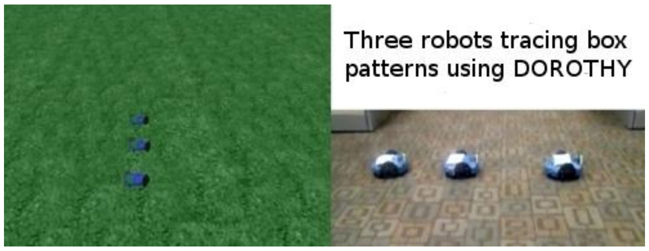


**Figure 5: Three fluke robots drawing a box in synchronization with their virtual representations.**

## 4.3 Curriculum Development and Pilot Runs

We have developed a curriculum that incorporates the scenarios described above in individual lessons that teach core computing concepts. These lesson plans can be viewed online: http://www.cs.ttu.edu/~smohan/Outreach/Dorothy.html

The curriculum consists of six lessons, each of ~1.5-2 hour duration, which can be distributed over a few days or weeks. The first lesson illustrates the importance of computing by discussing recent technological developments and logic puzzles. Instructors also demonstrate DOROTHY and its ability to control one or more robots. The second lesson has student teams interacting with robots through the virtual interface, e.g., generate graphical routines to control a robot's motion. Instructors demonstrate the need for sensor inputs from robots, and illustrate the use of basic variable assignment for generating motion patterns defined by user-specified parameter values. Students then work on recreating this demonstration. The third lesson introduces Boolean conditions and control structures (e.g., *if-else*) using scenarios such as obstacle avoidance. Students create appropriate graphical routines and execute them on real-robots. The focus of the fourth lesson is on iteration, which is motivated by posing challenge tasks such as repetitive pattern generation and obstacle avoidance. Student teams learn from demonstrations and create programs for similar tasks. Lesson five motivates the need for functions for program reuse and introduces basic image processing concepts. Students use these concepts for complex tasks such as locating bright spots in a room and visual object recognition. The final lesson discusses concurrent execution concepts and poses challenging tasks as projects, e.g., multiple robots performing a dance or searching for treasure. Student teams participate in a competition to complete these projects.

Different versions of the curriculum are used for middle school and high school students. For middle school students, the lessons focus on providing a general overview of computing and on developing basic problem-solving skills. Only the first three lessons are used and simple puzzles and tasks are assigned in the individual lessons. For high-school students, all six lessons are used to provide a more in-depth knowledge of core computing concepts such as problem solving, functions, iteration, control structures, abstraction and concurrent execution.

We are iteratively modifying DOROTHY and the associated curriculum based on pilot trials at a local middle school and high school. For instance, we conducted a workshop for students at a local high school that engaged students in projects (based on the first four lessons described above) in four 75 minute sessions spread over four weeks. A workshop was also organized for ~20 middle school students to use DOROTHY for interacting with flukes in three one-hour sessions. Furthermore, a three day workshop (total of seven hours) was conducted for ~20 middle school girls hosted on campus for the *Science: It's a Girl Thing* workshop. The workshops have been well-received by the students and their teachers, with feedback that has helped improve the tool and the curriculum.

## 5. SUMMARY AND FUTURE WORK

This paper described the DOROTHY educational tool that has enhanced the Alice 3D programming environment with the capability to control sensing and actuation commands on one or more mobile robots with on-board sensing capabilities. The modular architecture of DOROTHY introduces command dictionaries and robot handlers to enable synchronous (i.e., concurrent) and asynchronous operation of one or more virtual robots and their physical counterparts in the real-world. Bidirectional communication of data and commands between the tool and the robots results in more realistic simulations and adaptive behavior on the robots. The integrated environment has significant potential for use as a stimulating teaching tool for computational thinking since it integrates two of the most popular programming paradigms used at the K-12 level: 3D graphical programming and robotics.

The development of DOROTHY opens up many avenues of further research. For example, research is needed to better simulate how the software perceives the real-world using the robot's sensors. Another challenge involves synchronizing the speed of the virtual robots with that of the real robots. Currently, a virtual robot object can move at speeds that exceed the capabilities of the physical robot being commanded. Future research will also incorporate and test different wheeled and humanoid robots. Further integration of the more advanced sensing capabilities of the *erratic* robot (e.g., simple path planning, learning domain map, and learning object models) will enable the use of DOROTHY to illustrate stimulating real-world applications of computing.

One key direction for future research is focused on extensive evaluation of the tool and curriculum through workshops that engage middle school and high school students. The results of these studies will inform and guide the further development of curriculum materials for middle school and high school students. The work in [17] has already shown that an integrated Alice/robotics teaching approach can help engage beginning undergraduate students. DOROTHY is well suited for teaching core computing concepts at the middle school and high school level. In terms of educational research, the integration of 3D programming and robotics can be used to: (a) help teachers at the K-12 level teach (and enhance) computational thinking skills among their students; and (b) help students learn and apply these computing concepts to real-world challenges. Such studies may lead to a deeper understanding of effective ways to teach fundamental computing and programming concepts, further stimulating student interest in computing disciplines.

## 6. ACKNOWLEDGMENTS

This research was supported in part by NSF Grant No. CNS 1005212. Opinions, findings, conclusions, or recommendations expressed in this paper are those of the author(s) and do not necessarily reflect the views of NSF.

## 7. REFERENCES

[1] Alice (2009). Accessed May 2011 at www.alice.org.

[2] BPC (2010) NSF Broadening Participation in Computing Portal, www.bpcportal.org

[3] CPGE21 (2007) Committee on Prospering in the Global Economy of the 21st Century, *Rising Above the Gathering Storm: Energizing and Employing America for a Brighter Economic Future*, National Academies Press, 2007, 592pp.

[4] Craig, M., and Horton, D. (2009). Gr8 Designs for Gr8 Girls: a Middle-School Program and its Evaluation, *Proceedings of the Fortieth ACM Technical Symposium on Computer Science Education*.

[5] Dann, W., Cooper, S., and Pausch, R. (2008). *Learning to Program with Alice (2nd Edition)*, Prentice-Hall.

[6] Davis, J., Wellman B., Anderson, M., and Raines, M. (2009). Providing Robotic Experiences through Object-Based Programming (PREOP), *Proceedings of the 2009 Alice Symposium*.

[7] Doerschuk, P., Liu, J., and Mann, J. (2009). INSPIRED Computing Academies for Middle School Students: Lessons Learned, *Proceedings of the Richard Tapia Celebration of Diversity in Computing Conference*, Portland, Oregon.

[8] Greenfoot (2009). Accessed May 2011 at www.greenfoot.org.

[9] HCB (2010). Hispanic Computer Brigade, http://www.engr.sjsu.edu/hcb.

[10] IPRE (2010). The Institute for Personal Robots in Education, accessed May 2011 at http://www.roboteducation.org.

[11] IPRE Wiki (2010) The Institute for Personal Robots in Education Course Curriculum, http://wiki.roboteducation.org/Introduction_to_Computer_Science_via_Robots

[12] Kelleher, C., Pausch, R., and Kiesler, S. (2007). Storytelling Alice Motivates Middle School Girls to Learn Computer Programming, *CHI 2007 Proceedings • Programming By & With End-Users*, April 28-May 3, San Jose, CA, USA.

[13] Lauwers, T., Nourbakhsh, I., and Hamner, E. (2009). CSBots: Design and Deployment of a Robot Designed for the CS1 Classroom, *Proceedings of the 2009 SIGCSE Technical Symposium on Computer Science Education*, pp. 428-432.

[14] McWhorter, W. and O'Connor, B. (2009). Do Lego Mindstorms Motivate Students in CS1? *Proceedings of the 2009 SIGCSE Technical Symposium on Computer Science Education*, pp. 438-442.

[15] Parallax (2011). Accessed July 2011 at http://www.parallax.com.

[16] PCAST (2007) President's Council of Advisors on Science and Technology (PCAST), *Leadership Under Challenge: Information Technology R&D in a Competitive World*, Executive Office of the President of the United States, August 2007, 63pp.

[17] PREOP (2007). The University of Alabama, accessed July 2011 at http://cs.ua.edu/preop.

[18] RCJ (2010). The Junior League RoboCup Competitions, *www.robocup2010.org/competition_Category.php?c=4*

[19] Scratch (2009). Accessed May 2011 at www.scratch.mit.edu.

[20] Summet, J., Kumar, D., O'Hara, K., Walker, D., Ni, L., Blank, D., and Balch, T. (2009). Personalizing CS1 with Robots, *Proceedings of the 2009 Technical Symposium on Computer Science Education*, pp 433-437.

[21] Wellman, B., Anderson, M., and Vrbsky, S. (2009). PREOP as a Tool to Increase Student Retention in CS, *Journal of Computing Sciences in Colleges*, 2009.

[22] Wellman, B., Davis, J., and Anderson, M. (2009). Alice and Robotics in Introductory CS Courses, *Proceedings of the Richard Tapia Celebration of Diversity in Computing Conference*, Portland, Oregon, April.

[23] Wing, J. M. (2006). Computational Thinking, *Communications of the ACM*, 49, 3 pp. 33–35.