From 9 to 90: Engaging Learners of All Ages

Allison Sauppé, Daniel Szafir, Chien-Ming Huang, and Bilge Mutlu
Department of Computer Sciences, University of Wisconsin–Madison
1210 West Dayton Street, Madison, WI 53706, USA
{asauppe, dszafir, cmhuang, bilge}@cs.wisc.edu

ABSTRACT

This paper details the creation of a two-day computer science and robotics outreach course aimed at simultaneously engaging youth (children, ages 9–14) and senior (their grandparents, ages 55+) students. Our goal is to encourage enthusiasm for science and technology in students of all ages as well as provide practical instruction regarding common computer science concepts, including variables, loops, and boolean logic. To this end, we ground our course in the emerging field of social robotics, which enables the design of several multidisciplinary hands-on activities for students. We report on a four-year experience in the development of our course, which has been offered twelve times and involved over 210 youth and senior students. Our work presents a discussion regarding the challenges in designing a course for students from diverse ages, guidelines for creating similar courses, and a reflection on how we might improve our own class. The activities and project code developed for our course are available online as open-source resources.

Categories and Subject Descriptors

K.3 [**Computer and Information Science Education**]: Curriculum

Keywords

Outreach; Education; Curriculum; Social Robotics; Robots; Intergenerational Learning

1. INTRODUCTION

Computer science has emerged as a field with broad appeal and the potential for great impact across society. As such, there is an increasing interest on the development of tools that support computer science education and enthusiasm at younger ages, including software, such as Alice [7] and Scratch [11] that directly target youth, and hardware, such as Lego Mindstorms [2], Arduino [1], and Raspberry Pi [18], which encourage hands-on experimentation.

In addition to using such tools, youth computer science education might be further supplemented by engaging in simultaneous learning alongside their older familial mentors, such as siblings, parents, or grandparents. Programs with youth-mentor pairings in small, focused outreach courses are becoming increasingly popular and are now offered at several universities, including the University of Wisconsin, Michigan State University, Winona State University, University of North Texas, and Grand Valley State University. Such outreach courses represent a unique opportunity for encouraging computer science education in young and old alike, and thus offer a rich potential design space for the creation of new computer science curricula that inspire life-long learning. However, creating classroom experiences that tailor to a variety of age ranges simultaneously can also introduce new educational challenges due to discrepancies between young and old attention spans, prior technological familiarity, and topic interest.

In this paper, we share our experiences addressing such challenges during the creation and ongoing development of a two-day *Social Robotics* outreach course at a public university that offers instruction in computer science topics for children and their grandparents. The class has been offered twelve times over a four year time period and has instructed over 210 students regarding concepts integral to computing, artificial intelligence, robotics, and social interaction research. We report on the design of a variety of classroom activities and programming exercises (see Figure 1), which are available online for use in developing similar courses[1], our experience teaching the course, and conclude with an analysis regarding how our teaching approach might extend to include additional computer science topics, utilize additional platforms, and inform the development of similar courses in the future.

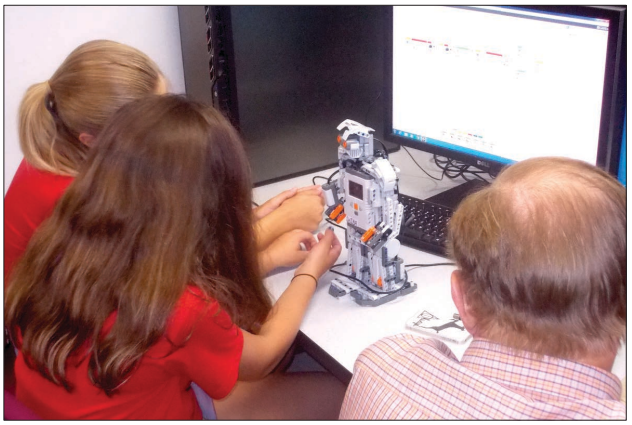

Figure 1: A grandfather and two of his grandchildren work together to program a Lego Mindstorms robot.

[1] http://hci.cs.wisc.edu/outreach/gpu

SIGCSE '15 Kansas City, Missouri USA

http://dx.doi.org/10.1145/2676723.2677248.

2. BACKGROUND

Although many students are introduced to computer science at the high school or college level, there is an increasing effort to adapt instruction to support learning in other age groups. Below, we discuss prior work that tailors computer science curricula, particularly those involving robots, to younger students. To situate our course, we also give background on the field of social robotics and describe our classroom environment, which is part of an initiative known as Grandparents University that caters to multiple age groups.

2.1 Teaching Diverse Age Ranges

In an effort to diversify the demographics of computer scientists, many programs have been created to foster an interest in computer science concepts in the K-12 age range. Tools and curricula have been made accessible for younger students in topics such as parallel programming [5], digital image processing [12], game development [15], and big data [4]. These courses aim to provide active engagement while adapting concepts for the less developed mathematical and cognitive skills of younger students [8].

To date, such adaptations in instruction have often been reserved for the K-12 demographic, while older generations have largely been ignored. Prior work has found that *computer shyness*—an aversion to learning about technology—is positively correlated with age [16], potentially increasing the difficulty of motivating senior students to invest in learning computer science and suggesting that engaging older generations might require modifying the tactics used with young children.

Prior work on *intergenerational learning* (IGL)—where participants from two or more generations work together in a learning environment—suggests that each generation has unique capabilities and perspectives to offer students in the other generation(s), enhancing the learning experience of each group [14]. IGL has additional social benefits by fostering the creation of meaningful intergenerational relationships, with each generation gaining an increased appreciation for other generations [14]. Given the unique challenges of teaching computer science to older and younger generations of students, IGL has the potential to leverage the strengths of each demographic to enhance the overall learning experience of the group, with each generation bolstering the weaknesses of the other.

2.2 Robotics in CS Education

Robots are increasingly being used as integral parts within computer science and engineering curricula. Educators and curriculum organizers cite that robots are beneficial to their programs by providing a hands-on approach to learning and tangible outcomes for students engaged in programming tasks [6, 9, 17]. Robotics curricula have been developed to augment introductory computer science courses to engage a more diverse audience [9, 17] and are also being created to attract younger students to the field [6].

While robots appear to be a promising tool in supporting the instruction of various computer science concepts, they are simultaneously becoming further integrated in applications throughout society, for instance acting as hospital aides, museum tour guides, and domestic assistants. As robots become increasingly prevalent, it is important to develop robots that behave and interact with humans in a socially acceptable manner [13]. While research fields such as human-robot interaction (HRI) and social robotics explore this issue, little work has been done examining how programming social robots might be used to convey traditional computer science topics to younger students. However, work in cognitive psychology suggests that the use of a robot contextualized amongst social cues might be a beneficial approach. *Embodied cognition* emphasizes the role of the body and its place in the environment in creating cognition, while *grounded cognition* emphasizes the role of external factors in shaping cognition [3, 10]. While differing they are viewpoints, both suggest that cognitive functions are inherently influenced by the body and connected to the world. Thus, the use of a robot contextualized in social behaviors, which has physical presence and familiar embodied action, might be a beneficial approach for learning computer science concepts.

2.3 Grandparents University

The curriculum we describe in this paper was developed for a two-day summer camp called Grandparents University (GPU) at the University of Wisconsin–Madison. GPU is sponsored by the university's alumni association, and is an opportunity for grandchildren and their grandparents to "major" together in a field of study. Each year, GPU offers several majors, ranging from *Textile & Design Studies* to *Limnology*. The concept of GPU has become popular in recent years, and multiple incarnations of the program now exist, both at other universities and through local clubs and societies. At our institution, grandchildren and grandparents spend a total of six hours in their major, three on each day of the camp.

In this work, we focus on the *Social Robotics* major, which the authors created and have offered twelve times since 2011. In our course, we teach children and their grandparents about how humans interact with each other and how we might imbue robots with similar behaviors to facilitate natural and intuitive interaction with users. For instance, we describe how humans use social cues, such as facial expressions and body language, to communicate, and how those same cues can be emulated on robots. We then instruct students regarding how to program several simple social behaviors on Lego Mindstorm robots. Our major targets children 9-14 years of age and their grandparents, with each course divided evenly between grandparents and grandchildren. We chose nine as our lower bound due to the math background needed to complete basic programming tasks. Our local GPU does not admit children older than 14, hence our upper bound. Although a *Computer Science* major concerning Scratch is also offered at GPU, it is not a prerequisite for our course and, thus far, it is rare for a student to have taken the Computer Science major prior to participating in our major.

While prior work has explored how computer science curricula and tools can be adapted for age ranges younger than the traditional college classroom, our work addresses designing curriculum for multiple age groups in a single class, such as in the GPU setting. Additionally, we provide the template for a social robotics outreach program, which offers a unique opportunity to engage students using a blend of computer science, robotics, and social sciences.

3. COURSE DESIGN

In this section, we describe the characteristics of our students and their associated challenges and opportunities, the objectives of the course, and our course design.

3.1 Challenges and Objectives

Our course targets students with two distinct age groups—grandchildren from 9-14 and grandparents from 55 and above. Grandchildren might be better with and willing to explore technology uses, however they may demonstrate shorter attention spans to focus on course materials and have more limited math and cognitive skills. On the other hand, grandparents have more life experience and cognitive capability to associate and process course materials, but may be less willing to experiment and try new technology. While these characteristics of our students introduce challenges in curriculum design and implementation, they present opportunities

for intergenerational learning, allowing grandchildren and grandparents to complement one another.

In addition to creating opportunities for children and grandparents to learn together, we have the following objectives for our course.

1. Expose students to the emerging field of social robotics
2. Introduce students to basic computer science concepts
3. Encourage students to learn from doing, participating, and exploring

We developed a two-day course to address our curriculum objectives for both grandchildren and grandparents. In the first day, instructors provide students with background knowledge of robotics, direct activities that develop real-world research skills, and encourage students to think about and explore the design space of social robots. In the second day, students put their newfound knowledge into practice while programming robots during hands-on exercises that teach introductory computer science concepts.

3.2 Day 1: Introduction to Social Robotics

We created a series of lecture topics and group activities on the first day of the course to familiarize students with the emerging field of social robotics. Our goal was to engage students about the topic, as well as leverage the cognitive capabilities of grandparents in transmitting that understanding to grandchildren. Lecture topics are loosely grouped into four high-level categories, detailed below.

3.2.1 Cultural and societal background

The class starts with an overview of how social robots, such as Furby toys and Keepons, have emerged in popular culture and society. Students then offer other robots they have encountered, either in popular culture or their own experiences, and what might make them appear more or less social. Common answers include R2-D2 and C-3PO from *Star Wars*, Rosie the robot maid from *The Jetsons*, Roombas, and the AIBO robotic dog toy. This discussion encourages students to think about the applications of robots, the difference between robots and other mechanical tools, why it is important for robots to have social knowledge, and the potential impact robots can have on culture and society. While children are often more familiar with current robotic technologies, grandparents can help their grandchildren contextualize robots' roles in society.

3.2.2 Design space of social robots

The class is then exposed to the design space of social robots. We utilize two activities to highlight the importance of social cues and appearance design in this area.

Social Cues: To introduce the class to the concept of social cues, we ask for a single grandchild to volunteer to help us. With the grandchild and instructor at the front of the room, we ask the grandchild to tell the instructor about their favorite movie. As the volunteer starts discussing the movie, the instructor acts to give the impression of boredom, such as looking away from the volunteer, yawning, or even using their cell phone. After a few moments, the instructor stops the volunteer and asks them if they thought the instructor was paying attention. When the volunteer says no, the instructor asks both the volunteer and the class what gave the impression of boredom. The instructor then asks the volunteer to resume discussing their favorite movie. This time, the instructor appears engaged by making eye contact, facing toward the volunteer, nodding and using appropriate backchannels (e.g., "uh-huh"), and asking questions to the volunteer about the movie. Again, the volunteer and class then discuss what impression the instructor gave and what social cues the instructor used to convey that impression. This activity introduces the class how we humans are wired to produce and interpret social cues and highlights implications of designing social cues for robots.

Appearance Design: The class is introduced to the importance of robot appearance and design in communicating functionality and completing tasks. Grandchildren work together with their grandparents to design a robot for a particular task of their choice, such as playing baseball, helping with homework, or doing the dishes. Grandchildren frequently choose their robot's task focus, helping to engage them. Grandparents then help their grandchildren focus on creating a design that addresses the robot's communication with humans when completing the task.

3.2.3 Design methods

After exploring the design space of social robots, we conduct three activities that engage the class while exposing them to real-world design methods used when developing social robots.

Ethnography: One of the foremost challenges in social robotics is understanding and defining the variant social behavior of the human population. For example, some people are more prone than others to nodding during a conversation. Observing human-human interaction using *ethnographic* data collection, where researchers study interactions *in situ*, is a typical approach to capture the variability in human behavior and interpersonal interaction (e.g., [13]). To let the class experience this research method, the instructor separates the class into groups of three to five people, with four people being ideal. Each group chooses a *speaker* and a *listener*, and the remaining group members are assigned the role of *observers*. All observers receive a slip of paper with their instruction: count the number of times the listener nods. The speakers then give a brief narration to the listeners on a topic of their choice for approximately three minutes (e.g., recalling a memory spent with their grandchild/grandparent). At the end of the allotted time, the instructor asks observers to share how many nods they counted. Usually, the number of nods varies from near zero to the teens, and the count often varies between observers in the same group, showcasing not only the variety of human behavior but also the difficulty translating it into objective phenomena. If the point has not already been raised, the instructor can ask observers what behaviors they considered to be a nod. Some participants will suggest that multiple up and down head movements in short succession constitute a single nod, while others will count each oscillation individually. This discussion helps the class understand the challenges involved in defining human social behavior, which can be difficult for multiple people to agree on, even with a single data point.

Affinity Diagramming: Affinity diagramming is a common design practice that aims to categorize data by forming logical groups. As a followup activity to the initial discussion of social cues, we conduct an affinity diagramming activity that allows the class to better understand the variety of social cues used by humans in day-to-day interactions. Groups have 5-10 minutes to brainstorm as many social cues as possible, writing each social cue on a separate sticky note. Grandparents help grandchildren expand their initial set of social cues (e.g., gazing) to more nuanced social cues (e.g., winking). When the groups are finished brainstorming, the instructor asks grandchildren to bring up sticky notes for each of the following categories in order: "face" (e.g., smiling, gaze, winking), "gestures" (e.g., "posture" (e.g., facing toward/away, slouching) and "vocal" (e.g., "uh-huh," "um"). As grandchildren categorize their social cues, the instructor can read aloud what cues are being posted, discussing the variety present. Once all the cues are posted, it becomes apparent that many of the social cues are placed under the "face" category. The relative distribution of cues amongst these four cat-

egories creates an opportunity for discussion about how humans direct the majority of their focus to others' faces during interactions.

Storyboarding: We teach the class how to use storyboards to design scenarios that integrate desired social cues, learned from previous activities, into human-robot interactions. Storyboards are an important tool that allows researchers and designers to prototype how cues might shape interactions at a high level. First, storyboarding is taught by examining a number of comic strips that have a range of social cues in them. Each group is then given five pieces of paper, allowed to choose their storyboarding scenario from a variety of options, and instructed to create their scenario visually using one frame on each piece of paper. For example, scenarios might describe a favorite memory the grandparent and grandchild share (potentially one described in the ethnography activity), or might involve further developing a particular use case for the robot they designed in the appearance design activity. Students are told to pay particular attention to how social cues are exhibited in the resulting interaction. Grandchildren and grandparents are then invited to share their storyboard with the rest of the class, pointing out why those chose particular social cues and how they might support the interaction in the scenario.

3.2.4 Robots in the wild

After gaining background knowledge about robotics, the rich design space of social robots, and research methods used in developing social robots, the class is provided opportunities to see social robots in action during a series of interactive demos. This in-person experience with social robots help students contextualize the concepts they learned from the lectures and activities into concrete outcomes.

Robot Demonstration: Each year, the class spends about an hour in our robotics lab, where the instructor and other affiliated graduate students demonstrate various robots, including humanoid robots, telepresence robots, and flying robots. Students are encouraged to think back to previous class discussions regarding potential use cases and how and why such robots may or may not appear social. The instructor discusses how each robot is being used in the lab's research, the studies each robot has been used in, and the central research questions of each study, which often directly pertain to concepts covered earlier in the day, such as modeling human behaviors, implementing them on the robot platform, and testing their effects in small interaction scenarios. The session includes a demo using a small humanoid robot that dances and plays a guessing game with students, foreshadowing the coding activities they will complete in the second day. Following the demonstration period, we offer an impromptu question-and-answer session between grandchildren and grandparents and lab members, allowing students to get a behind-the-scenes look at research activities.

3.3 Day 2: Programming a Social Robot

In the second day of class, we guide students through a series of five programming exercises that teach students basic concepts in computer science and how they can use these concepts to develop social robots. In the first three years of the class, all programming exercises were conducted in the Lego Mindstorms NXT 2.0 development environment. The fourth year used the newer, but similar, Lego Mindstorms EV3 environment. Both Mindstorms software use a visual programming paradigm similar to Scratch[2], allowing users to arrange programming blocks in a sequential or parallel fashion in their workspace. Students are provided a workstation and Lego Mindstorms NXT 2.0 robot, configured as the humanoid robot "Alpha Rex" (Figure 2), for these programming exercises. Grandparents and grandchildren work together on a single platform, allowing

[2]http://scratch.mit.edu/

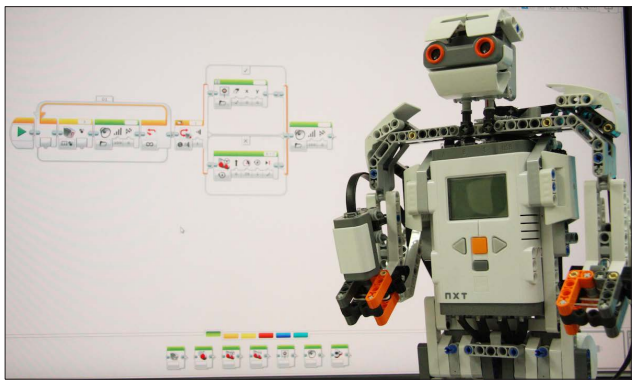

Figure 2: The Lego Mindstorms NXT 2.0 robot configured as "Alpha Rex." The eyes are an ultrasonic sensor and the robot's right hand is equipped with a color sensor. Behind the robot is the Mindstorms programming environment.

grandchildren to illuminate their grandparents understanding of programming. Grandparents also help to keep their grandchildren focused and augment the grandchild's problem solving capabilities.

The computer science concepts we teach include conditional statements, loops, data storage and retrieval (i.e., variables), random number generation, iterative development, and debugging. In addition, we also highlight building blocks for social robots involving input sensing and output generation, such as sound and robot motions. In the following paragraphs, we describe our designed programming exercises and how concepts in computer science and social robots were integrated into the exercises (Figure 3).

3.3.1 Program 1: Robot Voice

The first programming exercise teaches students the basics of the programming environment through the use of a sound block, which emits a chosen sound or phrase from the robot. The class first learns how to place a sound block in the environment, choose a pre-recorded sound for the block, and then execute the program. The class also learns sequential programming by arranging multiple sound blocks in a row, enabling the robot to speak a phrase.

3.3.2 Program 2: Motion Sensing

In social robotics, the ability to sense and respond to the environment is fundamental to interaction. We teach the class about making decisions based on sensor input through the use of an ultrasonic sensor, which is located in the eyes of the robot. We also introduce conditional statements. This exercise involves using the ultrasonic sensor to detect nearby movement. The goal is to produce a simple social interaction in which the robot says "hello" when a waving motion is detected. Building on the first program, if movement is detected, then a sound block that says "hello" is executed.

While such a program enables students to produce their first "interactive" robot, the algorithm only allows a single use. To enable the robot to repeatedly engage in the greeting interaction, a while loop is needed. To teach the concept of loops, we first create a new program with a single while loop that updates the display on the robot to show the number of iterations the loop has completed. When the class executes the program, the instructors ask them what number they see on the display and how that number is changing, eventually leading to the concept of a while loop and how quickly the program executes. We then extend the program from the previous exercise with a while loop, and have students wave multiple times to trigger the robot's greeting.

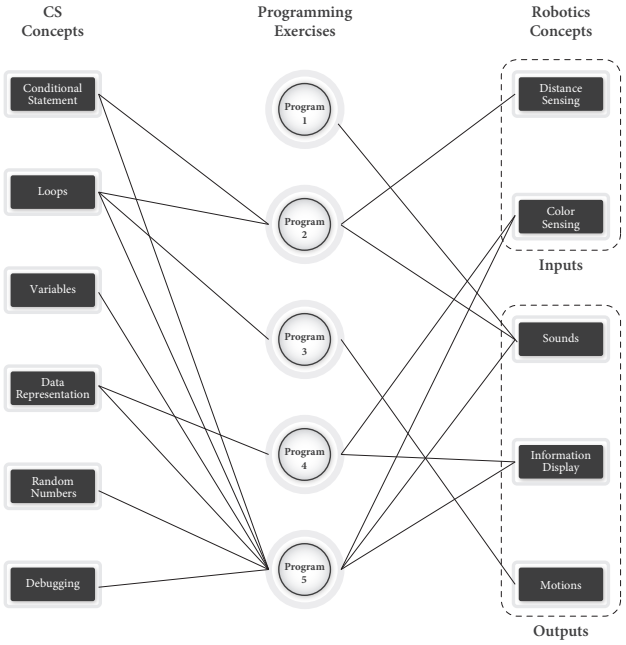

Figure 3: Programming exercises introduced students to concepts in computer science and social robotics.

3.3.3 Program 3: Robot Motion

Movements and gestures make a social robot more engaging and lively while enabling non-stationary tasks. In this exercise, we teach the class how to make the robot walk. The Mindstorms programming environment provides a walking block to allow programmers to achieve a fluid walking motion without the need to program parallel move actions. We also cover how to move individual motors, allowing students to move the arms to create simple gestures. Students thus gain mastery of the concepts of input and output, and how to use inputs to drive outputs.

3.3.4 Program 4: Data Representation

The Mindstorms programming environment creates abstractions for many of the data representations necessary in programming. For example, the Mindstorms robots have a color sensor that can detect six colors, which are stored as integers ranging from 1 to 6. We show students how to create a program that detects a color and displays the internal representation on the robot's screen. We then give students four colored balls each and let students experiment with the color sensor on their robot. Students can use the colored balls or often have colored pieces of paper, their name tags, mouse pads, dollar bills, and even their own shirts to try under the color sensor. Eventually, the class reaches the conclusion that each number represents a color, that colors are "rounded" to the nearest recognized color, and as a class creates the decoding map that translates each displayed number into a color. We then guide students in extending their color-recognition program to translate the numbers for each color such that, instead of numbers, a string representation of the color (the color name) appears on the display. This exercise further exposes students to conditional logic and environmental sensing while developing the notion of different types of data.

3.3.5 Program 5: Interactive Guessing Game

In the last exercise, we develop an interactive guessing game, similar to the demo students saw during the robotics lab tour in the previous day, which integrates the skills developed in previous programs and adds the concepts of variables and random numbers. In this game, the robot uses a random number generator to select one of the four available ball colors—red, yellow, green, and blue—and stores it in a variable. The human player makes guesses by placing one of the colored balls underneath the robot's color sensor, while the robot compares the data from the color sensor to the randomly chosen color stored in memory. If the player is correct, the robot says "good job." An incorrect guess elicits a "try again." Students usually encounter "bugs" in this exercise, such as the robot repeatedly saying "try again" while no guesses are being made, because it involves multiple components and coordination between them. Therefore, as part of this exercise, we teach the class basic debugging strategies and highlight the importance of developing small pieces of working, modular code before adding extensions.

After working through the above exercises, the class has been exposed to the core concepts of programming and how to implement these concepts in the Mindstorms programming environment. With the remaining time left in the class (between 45 and 60 minutes), grandparents and grandchildren have the opportunity to design and implement their own robot routine. The instructor is available for help with more advanced concepts the students would like to implement but were not covered in class, such as parallel programming. During the last 10 minutes of the class, each grandchild has the opportunity to show and explain their program to the rest of the class. Examples of past programs include a robot that uses the colored balls to make decisions about what to say to humans, dance routines, variations on the interactive guessing game, and even a robot "duel" that formed as a collaboration between two student groups.

4. LESSONS LEARNED

Over the course of teaching this program, we have found some approaches that work well, and others that could be improved. Additionally, there are some aspects of the program we would like to change, but cannot due to the limitations of our particular program. Below, we highlight and reflect on some of these issues.

4.1 Reflection

Reorganization of Activities: Currently, the course is structured to complete all the programming activities on the second day. However, experience and student feedback has shown that students enjoy working with the robots and would prefer to do some programming on both days. Ideally, the programming would be interspersed throughout the course to better engage students and to allow tighter loops between knowledge acquisition and implementation.

Movement is Engaging: While grandparents report being engaged with all the programming activities outlined in the prior section, those activities involving movement, in the form of walking, appear particularly popular with grandchildren. Additionally, nearly all student-designed robot routines end up incorporating movement in some way, highlighting the desire of students to physically "see something happen" as the result of successfully implementing an algorithm. The Mindstorms NXT 2.0 kits only have three motors, limiting the diversity of movement available to the robot. To enable a greater range of motions, the Mindstorms could be augmented with additional motors, or a different robot platform (alternatives discussed below) could be used.

4.2 Adoption for Other Programs

While our program is tailored to teach diverse age ranges about social robotics, the lessons presented can be adapted for other audiences. Below, we highlight ways our approach might be tailored for the needs of a particular program.

Targeting Other CS Topics: At a high level, we have designed our curriculum to engage a diverse age range by developing hands-on activities, tying activities to shared student experiences, and creating a framework where information is presented in an iterative fashion, where concepts are first introduced and then explored in greater depth. Many of these strategies might be adapted or extended for use in teaching other computer science topics. For example, to provide instruction on sorting algorithms, grandparents and grandchildren could order their shared relatives according to birthday, a hands-on activity tied to shared experiences. To teach parallel programming, a grandchild volunteer could come to the front of the class while a team of students each "manage" a "thread" controlling the volunteer (e.g., one student tells the volunteer what to do with their legs, another student controls the volunteer's arms, etc.).

Alternative Robot Platforms: The Mindstorms platform used in our course is commonly used in outreach. However, other robotics platforms are available with varying degrees of capabilities. Some examples include TinkerBots,[3] Cubelets,[4] and BetterBots.[5] Other commercially available platforms, such as Keepon, can be altered for more in-depth use by participants at a low cost.[6] For older or more experienced students, the Arduino robot platform may offer more creative freedom and advanced capabilities.[7]

5. CONCLUSION

We present the design of an outreach course aimed at simultaneously engaging young and senior students while encouraging enthusiasm for science and technology by providing instruction in computer science and social robotics. We highlight the challenges and opportunities involved in developing an intergenerational computer science course and present our curriculum developed over the past four years. In our experience, robots are an effective means of generating interest in all ages, provide a platform for hands-on instruction, and can generate a fruitful environment for discussion of technological progress between generations. Our experience has indicated that providing the lens of social robotics is both novel and timely. While developing material for both young and old students is a challenging endeavor, we have found it to be a tremendous opportunity to generate excitement and diversity in the field.

6. ACKNOWLEDGMENTS

We would like to thank the National Science Foundation award 1149970, the University of Wisconsin Alumni Association for their administration of GPU, and Shiyu Luo, who helped develop and teach our program.

7. REFERENCES

[1] R. Balasubramanian, Z. York, M. Doran, A. Biswas, T. Girgin, and K. Sankaralingam. Hands-on introduction to computer science at the freshman level. In *SIGCSE '14*.

[2] B. S. Barker and J. Ansorge. Robotics as means to increase achievement scores in an informal learning environment. *Journal of Research on Technology in Education*, 39(3):229–243, 2007.

[3] L. W. Barsalou. Grounded cognition. *Annu. Rev. Psychol.*, 59:617–645, 2008.

[3] http://tinkerbots.net/

[4] https://www.modrobotics.com/cubelets/cubelets-kt06/

[5] http://www.betterbots.com/

[6] http://probo.vub.ac.be/HackingKeepon/

[7] http://arduino.cc/en/Main/Robot

[4] P. S. Buffum, A. G. Martinez-Arocho, M. H. Frankosky, F. J. Rodriguez, E. N. Wiebe, and K. E. Boyer. CS principles goes to middle school: learning how to teach "big data.". In *SIGCSE '14*.

[5] C. Gregg, L. Tychonievich, J. Cohoon, and K. Hazelwood. EcoSim: a language and experience teaching parallel programming in elementary school. In *SIGCSE '12*.

[6] T. Karp, R. Gale, L. A. Lowe, V. Medina, and E. Beutlich. Generation NXT: Building young engineers with Legos. *IEEE Transactions on Education*, 53(1), 2010.

[7] C. Kelleher, R. Pausch, and S. Kiesler. Storytelling Alice motivates middle school girls to learn computer programming. In *CHI '07*.

[8] M. Knobelsdorf and J. Vahrenhold. Addressing the full range of students: Challenges in k-12 computer science education. 2013.

[9] T. Lauwers, I. Nourbakhsh, and E. Hamner. CSbots: design and deployment of a robot designed for the CS1 classroom. In *ACM SIGCSE Bulletin*, volume 41, 2009.

[10] B. Z. Mahon and A. Caramazza. A critical look at the embodied cognition hypothesis and a new proposal for grounding conceptual content. *Journal of physiology-Paris*, 102(1):59–70, 2008.

[11] J. H. Maloney, K. Peppler, Y. Kafai, M. Resnick, and N. Rusk. Programming by choice: urban youth learning programming with Scratch. In *ACM SIGCSE Bulletin*, volume 40, 2008.

[12] A. McAndrew and A. Venables. A secondary look at digital image processing. In *ACM SIGCSE Bulletin*, volume 37, 2005.

[13] B. Mutlu and J. Forlizzi. Robots in organizations: the role of workflow, social, and environmental factors in human-robot interaction. In *HRI '08*.

[14] S. Newman and A. Hatton-Yeo. Intergenerational learning and the contributions of older people. *Ageing horizons*, 8:31–39, 2008.

[15] A. Repenning and A. Ioannidou. Broadening participation through scalable game design. In *ACM SIGCSE Bulletin*, volume 40, 2008.

[16] N. Selwyn. Teaching information technology to the 'computer shy': a theoretical perspective on a practical problem. *Journal of Vocational Education & Training*, 49(3):395–408, 1997.

[17] J. Summet, D. Kumar, K. O'Hara, D. Walker, L. Ni, D. Blank, and T. Balch. Personalizing CS1 with robots. *ACM SIGCSE Bulletin*, 41(1), 2009.

[18] D. Wing and E. Meyers. Easy as Pi: Designing a library program to support computational thinking in preteens. *BCLA Browser: Linking the Library Landscape*, 6(3), 2014.