Exploring Computer Science through Autonomous Robotics

Zachary Henkel, Peggy Doerschuk, and Judith Mann
zmhenkel@tamu.edu, peggy.doerschuk@lamar.edu, judith.mann@lamar.edu

Abstract - **Declining student populations in the Computer Science field coupled with a growing demand creates the need for higher attraction rates. We examine new techniques for exposing high school students to the Computer Science field through the use of a Java-based autonomous robotics curriculum. This curriculum, designed by researchers to function as a means for conveying basic concepts during a limited engagement time, raises students with no prior knowledge of Computer Science to a level of comfort with both basic and advanced concepts within five sessions. Emphasis is placed on object-oriented design, behavior based programming, and autonomous operation of robots. The curriculum, structured into knowledge blocks which build on each other, provides students with clear goals and achievements during each session. Students work extensively with controlling robotic motors, sensors, and output devices through their own programs. Ultimately, students are asked to combine all of their learned knowledge by programming their robot to utilize behavior-based techniques to autonomously navigate a maze. Both quantitative and qualitative results have indicated significant knowledge gain, continued interest in the Computer Science field, and a great deal of enthusiasm from students participating in a program which utilized this curriculum.**

Index Terms – autonomous robotics, Computer Science attraction and recruiting, high school robotics, Java robots curriculum

INTRODUCTION

As society continues on a path which increasingly depends upon software development and maintenance, a declining population of undergraduate Computer Science enrollments sets the stage for an inevitable complication. The Computer Research Association reports a significant decline in interest in Computer Science as a major among incoming freshmen [1] beginning roughly in 2000. Furthermore, data from the 2006 - 2007 year shows a continued decline of 20 percent less than the previous year in Bachelor's degree production [2]. This decline has been widely attributed to the quick collapse of a large portion of computer related jobs during the "dot-com bubble burst". However, from 2006 to 2016 the demand for Computer Scientists is expected to grow by 37 percent, described as a field "growing much faster than average" by the United States Bureau of Labor and Statistics [3]. In an attempt to secure a sufficient amount of qualified Computer Scientists to satisfy this demand, Computer Science departments must focus their attention toward attracting and retaining students.

MOTIVATION

One of the primary issues relating to the lack of Computer Science majors manifests itself early on in the life of prospective students: students are uniformed about and not attracted to the field. Throughout primary and secondary education, Computer Science concepts are usually completely absent. A Program Summary report concerning the College Board's Advanced Placement Examinations for high school students indicates that only 20,532 students participated in the two Computer Science Exams in 2008. This number ranks in the lower half of all tests offered by the College Board AP Program, along with subjects related to specific non-English languages and Studio Art [4]. In addition to its low rank among other AP subjects, reports also indicate a decline in the number of students participating in exams in recent years [5]. This is alarming, as most other examinations offered have rarely seen declines in participation. Because of this absence of Computer Science influence in early education, students are less likely to be familiar with the field and its applications.

Additionally, low retention rates have been attributed to lack of computer experience prior to college years, with studies indicating students with no prior programming experience are at a disadvantage in earning a Computer Science degree [6, 7]. Computer Science departments must find a methodology for reaching out to high school students that can quickly offer an explanation of the field and its applications.

BACKGROUND

As object-oriented languages gain continued predominance in Computer Science programs, the "object-first" approach of teaching becomes increasingly important [9]. Our Java-based robotics curriculum employs this strategy while also harnessing the popularity of robots among students.

Other programs similar to ours focus more on competitive hardware design [10,12] or span multiple weeks [11]. Our work is closest to that of the Colorado School of Mines [14] but differs from theirs and others [13] in that our focus is based on object-oriented programming. All operations completed by students are a direct result of an

understanding of the object-oriented concepts embodied in the robot.

Our work is also unique in that we have developed a suite of instructional materials which can be easily utilized by any Computer Science group with access to an Intellibrain robot.

OVERVIEW

Our goals are to examine the effectiveness of a Java-based autonomous robotics curriculum which we developed to teach high school students core Computer Science concepts during a limited engagement time.

The Java-based autonomous robotics material reviewed in this paper was part of a larger student attraction effort by Lamar University's Increasing Student Participation in Research Development (INSPIRED) Program. INSPIRED is a National Science Foundation funded Broadening Participation in Computing Demonstration Project that is designed to increase participation of women and underrepresented minorities in computing. The material presented was developed in the Summer of 2007 and first tested on students in the Summer of 2008 at the INSPIRED Summer Robotics Academy for High School Students. A team of ten INSPIRED Computer Science students and faculty led the program that utilized these materials. Efforts were independently evaluated by an experienced faculty member of the Psychology Department who is well versed in assessing educational programs.

Our curriculum utilizes an off the shelf hobbyist Intellibrain robot to inspire students toward an understanding of basic Computer Science concepts. The robot hardware, detailed further in the Hardware and Programming Environment section, enables students to navigate the world around them, controlling sensors and actuators through object oriented Java programs.

The following sections describe the logistics and approach for scheduling sessions along with some insight on teaching style; outline the methodology and goals of the robotics curriculum; provide a detailed analysis of course components and exercise; examine the robotic hardware and programming environment; and highlight our experiences with building block motivation. The last three sections explain how students combined all of their newly gained knowledge into one final project; and review conclusions and future opportunities.

ANATOMY OF A SESSION

The curriculum, specifically designed with the idea of limited and segmented engagement times, strongly supports building block style motivation. Our pilot program spanned five days with two seventy-five minute time slots each day for the robotics portion of the program.

These sessions took place in the INSPIRED lab at Lamar University. Each student was equipped with a robot, laptop computer, personal workspace and mentor. The student workstations were arranged facing a presentation area.

Each robotics session consists of three basic parts: formal explanation of concepts, live demonstrations and hands-on activity. The formal explanation portion of a session involves the most critical thinking and personal engagement with students. It is during this time that students are challenged to consider the ideas that will be part of the day's sessions. For example, Session One begins by engaging students in a discussion of how a robot's input and output devices could work in harmony to complete a task. As the formal explanation continues, students are taught how they can achieve a certain task on their robot. This involves both high level conceptual dialog and practical code examination and explanation.

Live demonstrations focus around explaining the base functionality of some constructs to students through the means of interactive demonstrations. For example, students are asked to consider the idea of a for loop in Session Two. Rather than solely explain the theoretical operation of this structure, instructors engage in a trial and error exploration of the concept. Strong focus is given to "edge cases", as exposure to these cases ensures a more complete understanding. Additionally, live demonstrations allow the incorporation of higher level concepts in basic explanations. In our for loop example, the live demonstration leads directly into a discussion of variable scope. This style of teaching allows students to see the direct consequences of changes, rather than simply read about them.

Finally, hands-on activities urge students to apply their newly gained knowledge as soon as possible. The practicing of newly conceived knowledge within minutes of its conception brings with it amazing consequences. Not only do students understand the practicality of what they have just learned, but they also gain a higher ability to retain this knowledge for future sessions. The hands-on portions of the sessions also serve as a creative outlet for students, engaging their interest.

CURRICULUM GOALS & MATERIALS

Our curriculum's objective is to familiarize students with the core concepts of Computer Science in a quick and enjoyable manner. The Java language provides an easy introduction to object-oriented concepts, defunct of cryptic notation, declared pointers, and the burden of explicit garbage collection. Additionally, the Java language, coupled with the API provided by the robot manufacture, allows us to implement multiple threads with great ease. Programming robots in Java makes it easy to illustrate the concept of classes and objects because each component on the robot (sensor, motor, display, etc.) is an object. Robotics provide students with a way to see their code in action and express their creativity.

In addition to familiarizing students with core computing concepts, the curriculum also promotes familiarity with: object oriented design, behavior based programming, autonomous robotics operation, working with

sensors and actuators, redundancy, calibration techniques, compile-time errors, logic errors and problem solving through software.

The material we developed has been packaged into a bundle that consists of an instructor's manual, keynote slides for each session, student exercise files, solution files and maze building plans for the final project.

The instructor's guide consists of an outline of each day, complete with preparation guidelines. Additionally, the instructor's guide contains an extensively detailed commentary which corresponds with the keynote presentations for each session. Found by instructors to have the most value, the "Line By Line" solution code that is included with the instructor's manual explains in detail the solution of each exercise.

The resource CD consists of the keynote presentations in multiple formats, videos of the exercises to be completed, student template files, and solution files. Students complete the exercises by inserting key statements into the template files. This allows instructors and students to focus on key concepts without spending large amounts of time on syntax.

Having these materials in a distributable package allows us to train mentors and session leaders with great ease. Additionally, the "Line by Line" guides help stand-in mentors to quickly gain an understanding and steer students in the correct direction.

DIVISION OF CONCEPTS

Written with the assumption of students who have no previous knowledge of Computer Science concepts or computer programming, our curriculum strives to quickly establish the basics with students. The curriculum we have developed is structured into five main segments. Table I explains the basic goals of each main segment. Most importantly, our sessions are designed with a careful balance of information and hands-on applications to hold students' interest. Session One and Session Four merit explanation.

TABLE I
SESSION DIVISION

Session	Description
1	**Introduction** – Students are introduced to the ideas of computer programming, robotics and the fusing of the two.
2	**Control** – Students become true Computer Scientists, simplifying their lives with loops and conditions.
3	**Sensors**- Students learn to read from the various hardware sensors of their robot.
4	**Behavior** – Behavior based programming techniques are introduced and students learn to create behaviors they will use later.
5	**Maze Solving** – Students combine their knowledge from each session in order to have their robot autonomously solve an arbitrary maze.

Session One works heavily with encouraging students to think beyond their daily end user computer use. The instructor engages in a conversation about how a computer can perform the tasks a programmer specifies. Because time is limited, we stop at the level of modern day high level programming languages—not diving into the world of architecture. Session One explains how a program is essentially a set of instructions followed line by line.

Though it is not conventional, we introduce methods other than the Java main method immediately. We also discuss primitive data types, and the coupling of these types along with methods, to produce classes—from which objects are created. This discussion is infused with many practical examples, which is necessary because almost all components of the robot are represented as Java objects.

Though we mostly focus on identifying robotic parts as objects, we introduce the concept of objects using practical examples from everyday life. Our primary introduction, seen in Figure 1, describes a possible simplified schema for an object describing a cellular phone. This example also allows us to explore the idea of objects containing references to other objects as part of their data members.

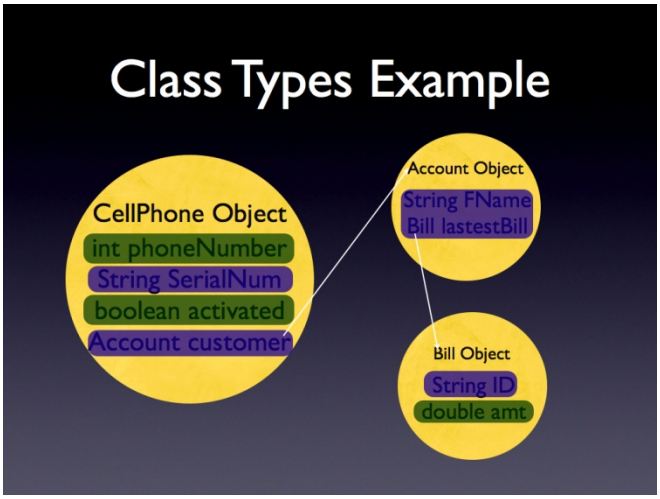

FIGURE 1
A SAMPLE SLIDE DETAILING TO STUDENTS THE NOTION OF OBJECT ORIENTED THINKING

Once students have a firm grasp on using objects, they have won much of the battle in instructing their robot. Because the Intellibrain robot provides access to its objects via static method call, we are able to divide the idea of using an object into three simple steps: library importing, object creation, and object control. Figure 2 shows a sample slide detailing to students how to use the display on their robot. Throughout the sessions students follow this simple process in order to obtain and control the objects they need to perform operations with their robot. The strong utilization of the library provided by Ridgesoft (examined further in the following section) teaches students to utilize code which has already been written in order to solve their own problems. Additionally, we also encouraged students to develop their own "motion methods" early on, in order to simplify common tasks like driving forward, backing up, turning and stopping. These techniques of code reuse are further examined in the Building Block Motivation section.

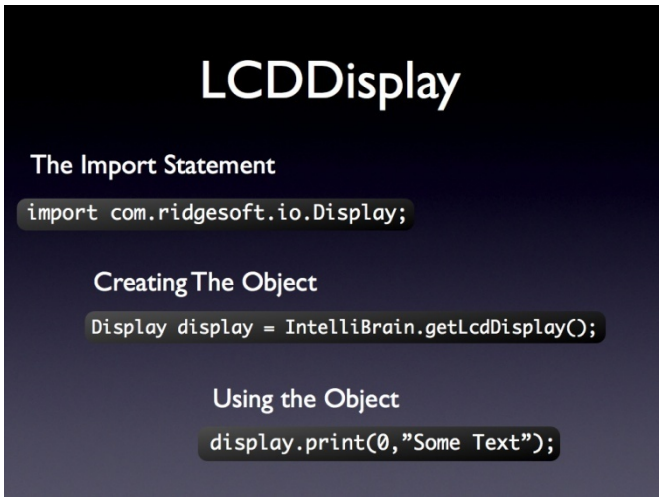

FIGURE 2
A SAMPLE SLIDE DETAILING TO STUDENTS HOW TO USE THE DISPLAY ON THEIR ROBOT.

Session Four is also worthy of analysis, as it introduces the concepts of behavior based robot programming. Behavior based robots are inspired by nature, which has many examples of seemingly intelligent tasks that are done by unintelligent creatures. Behavior based robots complete complex tasks without planning by combining simple behaviors. Already familiar with loops and conditions from Session Two and methods from Session One, in this session students must morph these ideas into what is required to build a behavior. Additionally, behavior based programming requires students to create their own class files, which will be used to instantiate objects. Previous sessions only require students to utilize objects which are provided to them by some other library. Behaviors created in this session include move forward, turn right at a corner, turn left at dead end, and stop at black line.

Students spend ample time analyzing the control flow of a program which uses a behavior arbiter. The arbiter selects which behavior to execute based on conditions in the environment and priority of the behaviors. Proper prioritization of the above behaviors can enable a robot to follow successfully a wall, a task necessary for maze solving.

HARDWARE AND PROGRAMMING ENVIRONMENT

The Intellibrain Robot, shown in Figure 3, was chosen as the base hardware for the implementation of this curriculum. Developed by RidgeSoft, the Intellibrain allows the attachment of many sensors and actuators. Additionally, this robot contains a Java Virtual Machine capable of running the code written by students. RidgeSoft also provides rigorous and helpful APIs, including support for behaviors, a behavior arbiter, and various sensors and effectors, as well as tutorials for programming the Intellibrain in Java. A sample of the API is shown in Figure 4; the full API is available at www.ridgesoft.com.

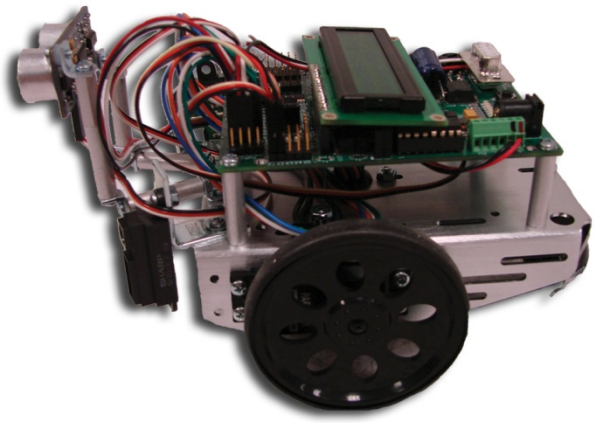

FIGURE 3
THE INTELLIBRAIN ROBOT

The robot, which runs on four AA batteries, connects via serial cable to RoboJDE, a development environment which runs on Microsoft Windows. This simple development environment allows for students to manage projects, write and compile code, and load programs onto their robots. In order to prevent constant disconnection and connection to robots via serial cable, we set up Bluetooth enabled connections between each student's laptop and robot via a third party adapter. Of the many sensors available for the IntelliBrain robot, our efforts focused on: sonar range finder, IR range finders and light sensing. Table II details the hardware used by students, grouped by session, along with a description of the hardware's abilities and common uses.

TABLE II
HARDWARE DIVISION BY SESSION

Session	Hardware Uses
1	Students are introduced to the robot's primary hardware. This includes servo motors, LCD display, and speaker.
2	Utilizing knowledge from hardware exposure from Session One, students expand their abilities to control already introduced hardware through software.
3	Students begin to work with infrared distance sensors, sonar distance sensors, and photo reflector light sensors . They are taught basic information concerning sonar pinging and IR angle reflection.
4	Utilizing behavior based techniques students strive to use their hardware knowledge to form ideal software behaviors.
5	Students demonstrate their mastery of understanding both hardware and software techniques, using all hardware to autonomously solve a maze.

An assembled robot with these sensors and Bluetooth connectivity is available for approximately $600. The robotic hardware may be obtained from www.ridgesoft.com. We worked to custom configure robots prior to student arrival for the most efficient sensor placement.

Constructor Summary	
protected	SonarRangeFinder() Constructs a SonarRangeFinder.

Method Summary	
float	getDistanceCm() Gets the distance to the nearest object
float	getDistanceInches() Gets the distance to the nearest object
abstract int	getEchoDelay() Gets the echo delay.
abstract void	ping() Generates a ping.

FIGURE 4
SAMPLE RIDGESOFT API - SONAR

BUILDING BLOCK MOTIVATION

Increasing the enthusiasm and enjoyment of our participants was of chief concern when creating this program. Though we had a limited engagement time available, we wanted participants to understand the fundamentally powerful idea of "beneficial laziness" in Computer Science.

As an example, we withheld the idea of loops and conditions for Session Two, for two reasons. The obvious reason, because it was incredibly difficult to fit all that we needed into Session One – but the not so obvious reason: we wanted students to appreciate the concepts which would come in the next session. During Session One, we asked students to make their robot beep a few times. To do this they had to write a line of code for each beep they wanted. Session Two introduced the loop, and instantly the beeping code became more dynamic and exciting. Suddenly, code which was yesterday's news became relevant again. Students began experimenting with more complex music making and in the process simplifying their code. We utilized motivations like this throughout our sessions in order to tease students along the learning path.

Another core concept we included was that of code reuse. Since emphasis was placed on object-oriented design early on in our teaching, students were empowered with the ability to write custom methods and then reuse them throughout the various sessions. During Session Two, students completed "motion methods" which simplified the common motions they needed to perform with their robots. These methods were not provided to them; rather they authored their own libraries and were able to use them throughout the various projects.

Results indicate that students were motivated by these building block motivation techniques. Most importantly, the hands-on exercises of Session One through Session Four worked to serve as the framework for the Session Five's final maze solving project.

FINAL PROJECT

As a final demonstration of knowledge, students were challenged with combining the exercises they had completed during previous sessions in order to successfully create a maze solving robot. Though several techniques are possible for solving a maze, wall following was chosen.

Figure 5 shows a top view of an Intellibrain robot making navigation decisions while solving a maze. As shown, multiple conditions read via sensors must be checked and prioritized in a limited amount of time in order to make correct navigation choices. Employing the technique of wall-following and harnessing the power of behavior-based decision making, students successfully accomplished this task.

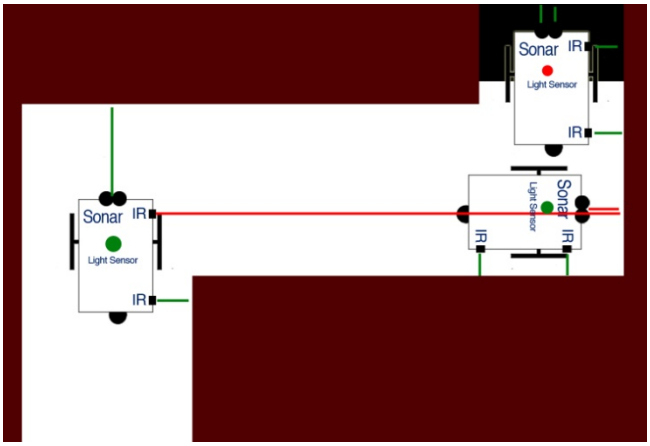

FIGURE 5
THE INTELLIBRAIN SOLVES A MAZE BY COMBINING BEHAVIORS MOVE FORWARD, TURN RIGHT AT CORNER, TURN LEFT AT DEAD END, AND STOP AT BLACK LINE. MULTIPLE STAGES ARE SHOWN HERE.

ASSESSMENT

An assessment of the impact of the robotics sessions documented significant knowledge gain, continued high interest, and a great sense of satisfaction among students.

In our 2008 pilot program the pre/post test content-based knowledge scores for the robotics portion of the academy rose from a pre-test average score of 2.14 to a post-test average score of 12.79. A t test for related samples was used to compare pre and post participation scores to determine if increase in knowledge was statistically significant. A t-value of 2.447 or larger is needed to be considered significant at the $p<.05$ level for the current research. The increase in knowledge was found to be statistically significant with a $t=5.10$, which was significant at the $p<.002$ level. Additionally, students indicated a continued high level of interest in the Computer Science field. Qualitative data, consisting of various interviews with participants, indicated a great deal of enthusiasm and a desire to return to a program of a similar nature.

Overall, the methods associated with this curriculum were found to be helpful and effective in explaining the core

concepts of Computer Science during a limited engagement time.

CONCLUSIONS

Our assessment indicates that the Java-based autonomous robotics curriculum that was developed successfully accomplished its goals of being an effective introduction to the field of Computer Science during a limited engagement time.

The unique combination of the Java programming language, Intellibrain hardware, session goals and teaching style has created a successful and very functional tool for introducing students with little to no previous Computer Science experience into the vast world of Computer Science.

We will continue to utilize, evaluate, and restructure this tool throughout the future.

ACKNOWLEDGEMENT

This work was supported in part by a National Science Foundation Broadening Participation in Computing Grant under Grant Number 0634288.

REFERENCES

[1] Vegso, Jay. May 2005, "Interest in CS as a Major Drops Among Incoming Freshmen" Computing Research News, http://www.cra.org/CRN/articles/may05/vegso.Accessed: 17 March 2009.

[2] "2006-2007 Taulbee Survey" 2008. Computing Research News. http://www.cra.org/statistics/survey/0607.pdf. Accessed 17 March 2009.

[3] U.S. Department of Labor. Bureau of Labor Statistics: Occupational Outlook Handbook (OOH). 2008 - 2009 edition. http://www.bls.gov/oco/ocos042.htm. 17 March 2009.

[4] College Board AP Program: Program Summary Report. 2008. http://professionals.collegeboard.com/profdownload/ap-data-2008-Program-Summary-Report.pdf. Accessed 17 March 2009.

[5] College Board AP Program: Exam Volume Change Report. 2008. http://professionals.collegeboard.com/profdownload/Exam_Volume_Change_11-3.pdf. Accessed 17 March 2009

[6] Morrison, M. and Newman, T. S. 2001, "A study of the impact of student background and preparedness on outcomes in CS I," SIGCSE Bull.33, 1 (Mar. 2001), 179-183.

[7] Hagan, D. and Markham, S. 2000, "Does it help to have some programming experience before beginning a computing degree program?" SIGCSE Bull.32, 3 (Sep. 2000), 25-28.

[8] Moskal, B., Lurie, D., and Cooper, S. 2004, " Evaluating the effectiveness of a new instructional approach," in Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education (Norfolk, Virginia, USA, March 03 - 07, 2004), SIGCSE '04. ACM, New York, NY, 75-79.

[9] Cooper, S., Dann, W., and Pausch, R. 2003, "Teaching objects-first in introductory computer science," in Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education (Reno, Nevada, USA, February 19 - 23, 2003), SIGCSE '03. ACM, New York, NY, 191-195.

[10] Cannon, K. R., Panciera, K. A., and Papanikolopoulo, N. P., 2007, "Second annual robotics summer camp for underrepresented students," in Proceedings of the 12th Annual SIGCSE Conference on innovation and Technology in Computer Science Education (Dundee, Scotland, June 25 - 27, 2007). ITiCSE '07. ACM, New York, NY, 14-18.

[11] "2008 UAB Computer Science Summer Camps," 2008, UAB, http://www.cis.uab.edu/programs/camps/. Accessed 17 March 2009.

[12] "Robot Camp" 2009. Union College. http://doc.union.edu/Robotcamp/. Accessed 17 March 2009.

[13] "Robotics Camp" 2009. ASU., http://asusrl.eas.asu.edu/srlab/research/roboticscamp/index.html. Accessed 17 March 2009.

[14] "Robostyle," 2009, Colorado School of Mines.http://inside.mines.edu/Outreach/Cont_Ed/courses/robo/index.html. Accessed 17 March 2009

AUTHOR INFORMATION

Zachary Henkel, Computer Science Undergraduate Student, Texas A&M University, zmhenkel@tamu.edu

Peggy Doerschuk, Professor of Computer Science, Lamar University, peggy.doerschuk@lamar.edu

Judith Mann, Assistant Professor of Psychology, Lamar University, judith.mann@lamar.edu