First Year Student Team Projects Using MATLAB

Kathleen Ossman, Gregory Bucks
University of Cincinnati, kathy.ossman@uc.edu, gregory.bucks@uc.edu

Abstract **- This paper describes two team projects developed for a two semester sequence of courses entitled Engineering Models I and II designed for all first year students majoring in engineering and engineering technology at the University of Cincinnati. In 2012-2013, the sequence was taken for the first time by all first year students in the College of Engineering and Applied Science. The courses apply fundamental theory from math and science courses to relevant engineering applications chosen from a variety of disciplines. MATLAB® is introduced and progressively developed as a programming tool to enable students to explore engineering concepts, to investigate solutions to problems too complex for hand solutions, to analyze and present data effectively, and to develop an appreciation of the power and limitations of computer tools. The first team project occurred during the last four weeks of Engineering Models I in the fall semester. Teams were required to develop a game or a set of games using MATLAB®. At this point in the sequence, students had basic programming skills but very little exposure to the graphic capabilities of MATLAB®. To make the project more engaging, several graphical tools were created by the instructors to allow the students to make their games visually interactive. The second team project occurred during the last four weeks of Engineering Models II. Each team was required to design a GUI in MATLAB® that could serve as an effective and engaging teaching tool for a topic that they learned about in one of their first-year courses. Students created GUIs on a diverse set of topics including differentiation, integration, Taylor series, organic chemistry, statics, projectile motion, and circuit analysis.**

Index Terms - First year engineering courses, MATLAB® programming, Problem Solving, Team programming projects.

INTRODUCTION

One of the challenges in engineering education is to convince students that there is a critical connection between the topics covered in their mathematics and science courses and their future engineering courses. Froyd and Ohland [1] reviewed the literature on integrated engineering curricula; they pointed out that most engineering curricula provide for sound foundations in mathematics and science, and anticipate that students will connect concepts from mathematics and science to the practice of engineering. However, there is evidence to indicate that the desired connections have not been made. Retention rates of first-year students are low, students see little connection between engineering and mathematics and science courses, and many students lack the ability to apply concepts from mathematics and science in engineering contexts. Cui et al reported similar results regarding students' transfer of learning from calculus to physics [2]. They found that solving calculus problems did not help students to solve isomorphic physics problems; students had difficulty setting up calculus-based physics problems, especially when identifying appropriate variables and limits of integration.

A second challenge in engineering education is teaching problem-solving skills. Many first-year engineering students are comfortable with the concept of exercise solving which only requires them to mimic examples provided by the instructor. However, synthesizing and applying concepts to solve a problem that is dissimilar to problems encountered before is problematic. Many authors have reported on the poor problem-solving ability of students. Heller, Keith, and Anderson [3] suggested that many physics students regard problem-solving as independent of physics concepts; they claim to understand the concepts but can't solve the problems. Many students also regard specific mathematical solutions to be the physics of interest; those students claim to understand the examples in textbooks but can't solve test problems because they are "too different." Woods et al [4] reported that many engineering students could not solve problems if the wording or context of the problem was changed. They also could not synthesize information from various sources to solve industrial problems.

A promising avenue to explore in order to foster the development of problem-solving skills and to bridge the content areas of mathematics, science, and engineering is through computing [5]. Given the extent to which computers have permeated the engineering design process, our engineering students must develop strong computing skills in addition to the traditional disciplinary skills. This sentiment has been echoed by many, including the National Academy of Engineering, who identified computing skills as one of the attributes required for future engineers in their Engineer of 2020 report [6]. Computing affords instructors the ability to introduce "hands-on" projects and activities early in the engineering curriculum while requiring little disciplinary knowledge on the part of the students and no additional materials. Hands-on projects and activities have been shown to increase student motivation and interest in course content and improve retention [7]-[8]. Through

computing, instructors can bring together concepts and ideas from mathematics, science, and engineering and allow students to interact with them, helping to form the mental connections necessary for more expert-like understanding [9]-[10].

In addition, many first-year engineering courses are tasked with developing the soft skills (communication, teamwork, etc) required by ABET and necessary to perform well as an engineer. Computing can also play a role in developing these skills. There has been extensive interest recently in the use of pair programming, which brings together pairs of students to work on solving complex computing problems [11]. Pair programming has been shown to increase student performance, motivation, and confidence [12]-[13]. These benefits extend to larger groups as well [14].

In the fall of 2012, the University of Cincinnati converted from a quarter system to a semester system. This conversion provided an ideal opportunity to review the first-year curriculum for the engineering and engineering technology students and make changes to help improve retention and performance of students in the College of Engineering and Applied Science (CEAS). The college faculty agreed on an almost common first year (Table I), which would include a one year sequence called Engineering Models to try to address the issues of poor problem-solving skills and the lack of connectivity between mathematics and science courses and later engineering courses. The Engineering Models sequence was developed and piloted over the two year period preceding the semester conversion. In 2012-2013, the sequence was required for all 800 first-year CEAS students.

TABLE I
FIRST-YEAR CURRICULUM

Fall Semester	Spring Semester
Engineering Models I	Engineering Models II
Engineering Foundations	Discipline Specific Engineering Course
Chemistry I	Physics or Chemistry II
Pre-Calculus or Higher	Calculus I or Higher

Engineering Models I and II is a two semester sequence of interdisciplinary courses in which students apply fundamental theory from algebra, trigonometry, calculus and physics to relevant engineering applications chosen from a variety of disciplines. MATLAB® is introduced and progressively developed as a programming tool to enable students to explore engineering concepts, to investigate solutions to problems too complex for hand solutions, to analyze and present data effectively, and to develop an appreciation of the power and limitations of computer tools. Special attention is given to graphical visualization of concepts and to numerical approximation techniques and the errors associated with approximations. The course objectives are:

(1) To explore the application of algebra, trigonometry, and calculus to various engineering disciplines,
(2) To learn the fundamentals of programming and good programming practices and utilize these skills to solve numerical problems and create numerical algorithms with MATLAB®,
(3) To develop good problem-solving skills by applying problem solving strategies to a variety of engineering problems, and
(4) To cultivate effective team-work and communication skills through lab work and design projects.

This paper focuses on the team projects in Models I and Models II. In Models I, student teams developed a game or a set of games. In Models II, student teams created a graphical user interface (GUI) that could serve as an effective teaching tool on a topic from one of their first-year courses or their chosen discipline. A description of each project, the preparation materials and pre-project lab exercises developed for the students, methods of assessment, and samples of student projects are included. In addition, student survey results and observations by the authors are included.

ENGINEERING MODELS I: GAME PROJECT

I. Project Description

The first team project occurred during the last four weeks of Engineering Models I in the fall semester. Teams were required to develop a game or a set of games using MATLAB®. Most teams consisted of three students but there were some teams of two and a few teams of four students. In order to accommodate a very wide range of programming ability and interest at this point in the course, teams were allowed to choose a set of simple games or a single more complicated game to achieve a total of four complexity points. A list of game suggestions along with complexity points is shown in Table II. Teams were required to choose at least one game with a complexity rating of two or higher. Teams were also allowed to propose their own games along with a justification for complexity level. The authors programmed most of these games prior to launching the project in order to decide on complexity points for each game.

TABLE II
COMPLEXITY POINTS FOR GAMES

Points	Game
1	Simple Dice Games (Craps, Over/Under Seven)
2	Simple Card Games (War, Go Fish, Memory)
2	Hangman
2-3	Solitaire – depending on complexity of gameplay
3	Black Jack
3	Master Mind
3-4	Connect Four – higher points for smart computer player
4	Othello
4	Yahtzee
4	Euchre
2-4	Adventure – depending on monster placement, number of levels and complexity of scoring
3-4	Battleship – higher points for smart computer player

II. Materials Developed for Students

At this point in the sequence, students had some basic programming skills (conditional statements, loops, and arrays) but had no exposure to the graphic capabilities of MATLAB® other than basic types of plots. In order to make the project more engaging for the students, several graphical tools were created by the instructors to allow the students to make their games visually interactive. The graphical tools included several popular game boards such as Connect Four, Master Mind, Othello, Adventure, and Battleship as well as decks of cards and dice. These tools were provided as .mat files accompanied by word documents describing the MATLAB® commands needed to display and update the various boards as gameplay occurred. For example, the Connect Four file consisted of a 6x7 cell array game board along with images of a red chip and a black chip that could be plugged into the board as the game proceeded. Images from this file are shown in Figure 1.

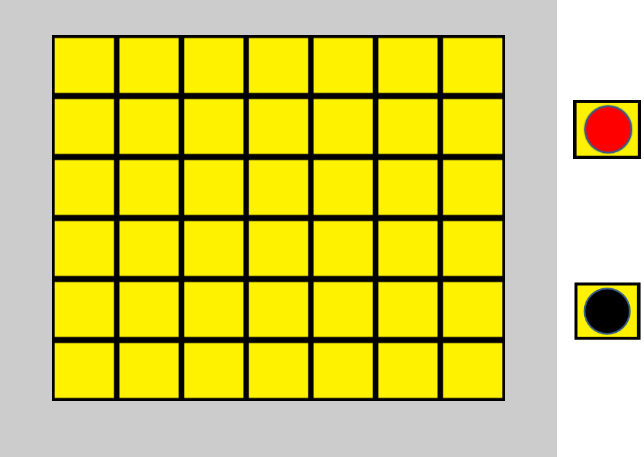


FIGURE 1
IMAGES FOR CONNECT FOUR

The week before the teams began working on the project, students were provided with a video that described all of the resources available for various games and also demonstrated several of the games. In recitation, students formed teams and completed a warm-up exercise, a simple two-player game of Tic-Tac-Toe, in order to become familiar with displaying and updating a game board, and creating a 3x3 numerical array to track player moves, determine allowable moves, and declare a winner.

III. Assessment

Teams worked on the games during recitation for the first three weeks of the project. At the end of the first and second week, the team leader was required to submit a progress report detailing work completed, work remaining, and any difficulties the team was encountering. In the final week, teams demonstrated their games to the instructor, teaching assistants, and all other students in the class. Each team was required to submit a final report and all game files. In addition, each student was required to submit a peer evaluation in which he/she evaluated his/her own contributions to the project and the contributions of the other team members. The project counted 15% of the course grade. Table III shows the rubric for grading the project.

TABLE III
GRADING RUBRIC FOR TEAM PROJECT (FALL)

Points	Criterion
5	Progress Report #1
5	Progress Report #2
40	Meets Complexity Point Requirements
5	Games are User Friendly
10	Used Good Programming Practices
10	Final Report
5	Oral Presentation (Game Demonstration)
5	Submitted Peer Evaluation Form
15	Individual Score

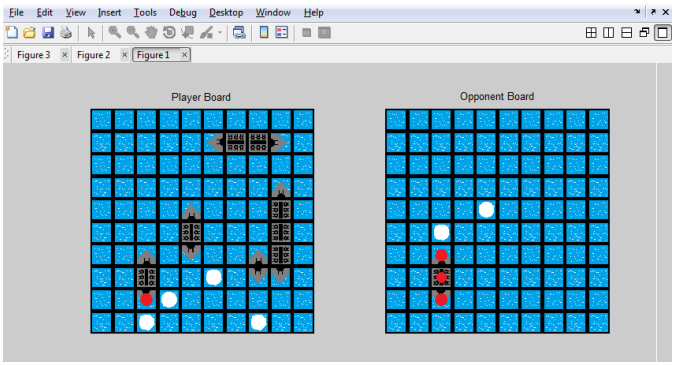
IV. Student Projects

Many of the student teams opted to program a set of simple games. We had quite a few hangman and war demonstrations to sit through. However, several teams chose more complicated games. In Dr. Ossman's three sections, 30 of the 64 teams chose to include a game with a rating of three or four complexity points. In Dr. Bucks' two sections, 30 of the 40 teams chose games with complexity levels of three or higher.

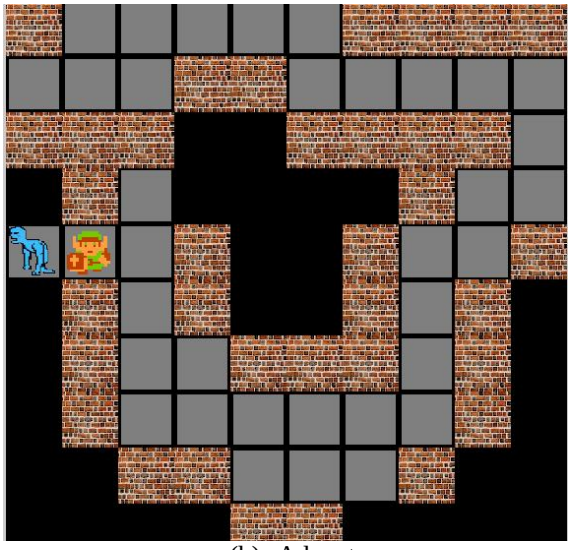
Battleship was a very popular choice. A screenshot of the graphical tools provided for Battleship is shown in Figure 2(a). Some teams opted to have the computer player shoot randomly but several teams incorporated intelligence on the part of the computer and added explosion sound effects. A few teams programmed multiple complexity levels (easy, normal, insane, chaotic, and absurd) and even added background music to accompany the game.

Adventure, Figure 2(b), was also a popular choice for many of the teams. Most of the teams doing Adventure chose to throw out the board, player, and monster graphics provided and incorporate their own characters instead. One team even went so far as to write a background futuristic story about a technology spy whose girlfriend was kidnapped by terrorists and created seven levels of the game which the player had to pass through to save his girlfriend. This team also composed their own background music.

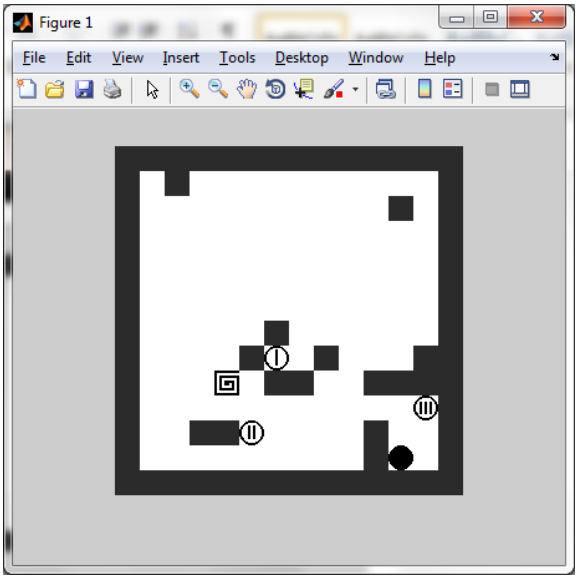
Another team studied the graphical tools provided and went on to create their own puzzle game shown in Figure 2(c). The player was required to manipulate boulders (circles with roman numerals) through a maze to an escape hatch while avoiding black holes. Seven different levels of difficulty were created and players beating a level received a code which allowed them to proceed to the next level of difficulty.



(a) Battleship



(b) Adventure



(c) Puzzle Game

FIGURE 2
IMAGES FROM SELECTED GAME PROJECTS

ENGINEERING MODELS II: GUI PROJECT

I. Project Description

The second team project occurred during the last four weeks of Engineering Models II in the spring semester. Each team was required to design a GUI in MATLAB® that could serve as an effective and engaging teaching tool about some topic that they learned about in one of their first-year courses. A GUI program has a modular structure; that is, each pushbutton, slider, pull-down menu, text block, and radio button panel added to the GUI results in a separate callback function within the program that needs to be coded. This modular structure allowed students to divide the programming tasks among the team members. In addition, students had the opportunity to be creative both in terms of selecting a topic to cover and in choosing which objects to use for their GUI to create an effective and engaging teaching tool for their topic. Most teams consisted of three students but there were some teams of two and a few teams of four students.

II. Materials Developed for Students

In recitation the week before the project started, all students participated in an instructor-led tutorial in which they created a simple GUI using GUIDE in MATLAB®. The instructor explained how to write code to get and set various properties of the objects. Students were also provided with a PowerPoint presentation that explained how to create a GUI using GUIDE, how to write code for the variety of objects available, and also included sections on frequently asked questions and common errors.

III. Assessment

For the first three weeks of the project, teams worked on the GUIs during recitation. A few days before the second week, the team leader was required to submit a progress report detailing what topic the team had chosen and how the work would be divided up among team members. In the final week, teams demonstrated their GUIs. Each team was required to submit a final report and all program files. In addition, each student was required to submit a peer evaluation in which he/she evaluated his/her own contributions to the project and the contributions of the other team members. The Teaching Assistants also evaluated each student based on their participation in and contribution to the project. Projects were evaluated by the instructors based on functionality, ease of use, effectiveness of the GUI as a teaching tool, creativity, and appearance. The project counted 20% of the course grade. Table IV shows the rubric for grading the project.

TABLE IV
GRADING RUBRIC FOR TEAM PROJECT (SPRING)

Points	Criterion
5	Progress Report #1
15	Functionality of GUI
15	Engagement/User Friendly
15	Complexity
15	Creativity and Appearance
5	Demonstration of GUI
15	Final Report
15	Individual Score

IV. Student Projects

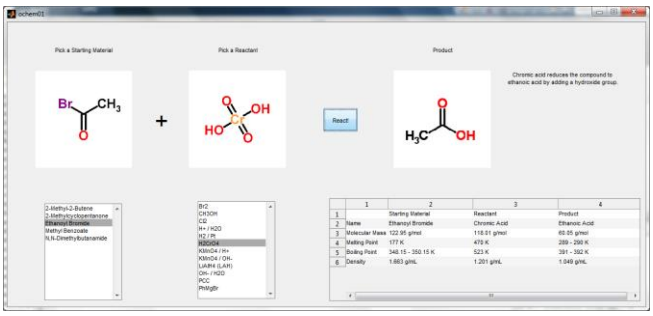
Students produced GUIs on a very diverse set of topics including differentiation, integration, Taylor series, organic chemistry, statics (beam loading), projectile motion, and simple circuit analysis. There was one team of eleven students that created a "Freshmen Survival Guide". This team broke into subgroups and covered topics from five of their freshmen courses: calculus, statics, MATLAB® programming, chemistry, and Solid Works.

Figure 3 includes screen shots from some of the GUI projects. The organic chemistry GUI allows the user to choose one of five possible starting materials and one of twelve possible reactants. Clicking on the React button will either produce an image of the product of the reaction and a table of properties for the materials or will indicate that the materials do not react and explains why not.

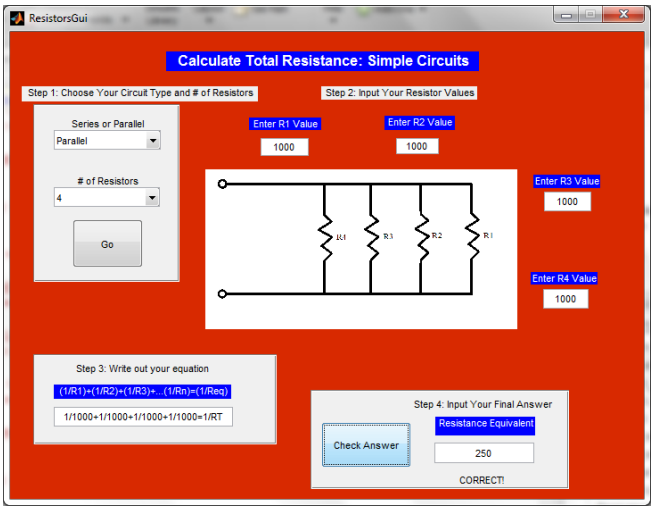
The Circuit GUI shown in Figure 3(b) allows the user to choose up to five resistors in series or parallel and specify values for each resistor. An image is produced based on the user's selection. The user is then prompted to calculate the total resistance and is able to check his/her answer.

The ALGO-RACE GUI shown in Figure 3(c) allows the user to select three different sorting algorithms from a set of five choices. The user also selects the maximum size of the vector to be sorted as well as characteristics of the values in the vector (random, few unique, sorted ascend, sorted descend). The GUI will then display the sorting process to the user and time the process for the three algorithms chosen.

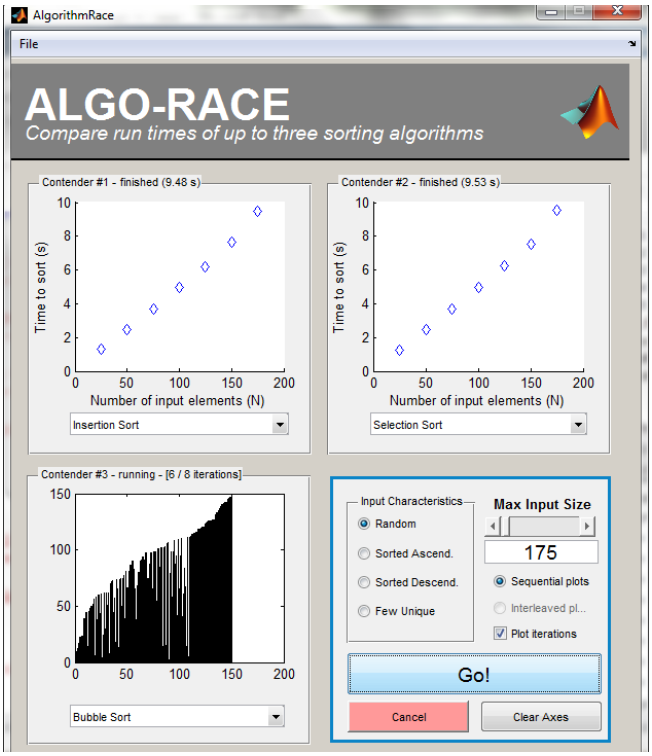
The Integration GUI shown in Figure 3(d) allows the user to pick one of six common functions and a set of limits. Based on the user's choice, the integral of the function is computed, the area under the curve is plotted, the volume of revolution is computed, and a 3-d graph of the solid of revolution is plotted.



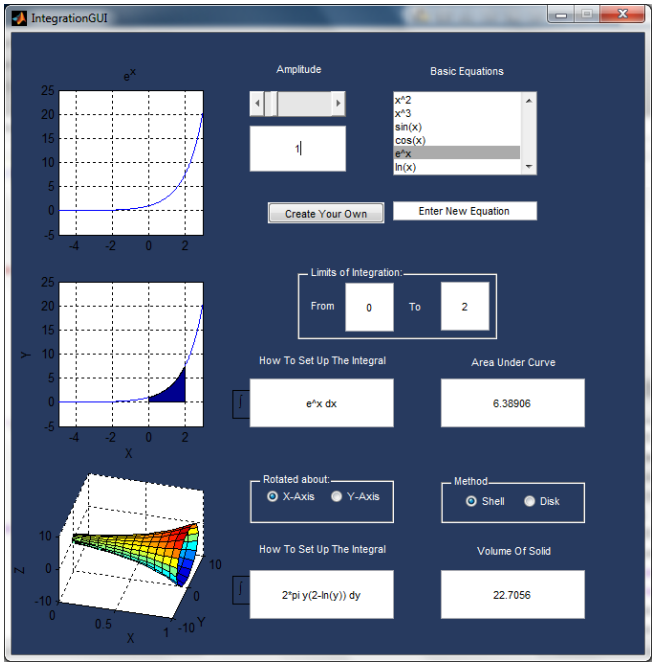
(a) Organic Chemistry



(b) Simple Circuits



(c) Sorting Algorithms



(d) Integration Concepts

FIGURE 3
IMAGES FROM SELECTED GUI PROJECTS

STUDENT COMMENTS

At the end of Engineering Models II, a survey was deployed using Blackboard. 443 students (65%) participated in the survey. Approximately one third of these students indicated that the team projects were what they enjoyed most in the sequence of courses. 18 students mentioned that they would have liked more preparation for the GUI project. Here is a sample of student comments about the projects:

- I enjoyed doing the GUI's because it allowed you to make a physical app in a way that is easy to learn how to make and create other concepts that were shown to us throughout the semester
- I enjoyed the projects. I thought it was very cool to actually apply our knowledge into a functioning product.
- I really enjoyed it all, but my favorite was probably the final projects, using mat files and GUIs to make our programs come to life. With the other programs, it would be just text asking for more text and computing even more text, but then in the projects, we made games. We made moving figures. Heck, in our GUI we made a spinning wheel. It really made clear everything that we've been learning, and gave a taste of the real-world applications.
- I most enjoyed the end of the semester projects because you got to create something that you wanted to and design every aspect of it.
- I enjoyed learning about GUI's because it provides a visual realization of the coding in a user-friendly setting. I also really enjoyed just the general process of writing codes and making programs do the work for the user, especially coming up with more efficient, user-friendly, less cluttered ways.
- Both projects were enjoyable. Having very little background in coding or any programs that operate like MATLAB, it was very beneficial for me to further develop my skills by working with those that have greater experience than me.
- On top of enjoying the general subject, I really enjoyed the final projects for both courses. They were a great test of my skills and knowledge of MatLab, and I enjoyed applying everything I've learned into one program. Furthermore, it was really satisfying to finish such projects and know that we created really awesome programs by ourselves from scratch.

INSTRUCTOR OBSERVATIONS

Both of the projects required students to function effectively as a team: brainstorming ideas, utilizing the talents of each member of the team, communicating with one another, and distributing and scheduling the work. Designing an effective team project where all students in the group participate is challenging because the tendency is for the strongest member(s) in the group to complete most of the work with little input from the rest of the group. The modular structure of the GUI allowed teams to divide tasks among the team members. For the game project, several teams chose to do a set of games and were able to divide the development work up by game.

The team projects required far more programming than any of the recitation and homework assignments from the two semesters. Although some students complained about the difficulty of the assignments, very few had complaints about the projects. Games and GUIs are viewed as fun by the students and therefore worth the extra effort. Students seemed much more willing to experiment with code and explore MATLAB® to find functions that would accomplish what they wanted in their projects.

The open-ended nature of the project allowed the students to express their own ideas and make the project their own. With most of the recitation and homework assignments, students were presented with a problem to which they had to find the "right" solution. Since the course caters to all engineering majors in the college, many questions on assignments did not relate directly to a student's chosen major. With the projects, students had the freedom to choose their own topic, making it much more personal and motivating them to go above and beyond the minimum requirements for the project.

One of the biggest challenges of these projects for the instructors and teaching assistants was the sheer number of teams and the diversity of projects each team was working on. Allowing students to choose from a long list of games

(or propose their own) and to select their own topic for the GUI made each team's project pretty unique. Dr. Ossman had 64 teams working on games in the first semester and 60 teams designing GUIs in the second semester. Dr. Bucks had 40 teams working on games in the fall semester and 61 teams designing GUIs in the spring semester. This represented only about half of the students enrolled in the sequence.

For the gaming project, there were a few students that found (and copied) code on-line, mainly through the MathWorks Central File Exchange. This was pretty easy to spot and deal with. Students that used the graphical tools that we provided could not use pre-written code. In the future, we may require students to use the provided graphical tools. In retrospect, it may have been better for us to offer a smaller set of games as choices since we will likely need a different set of choices next year.

FUTURE WORK

The retention data for the first year students from last year will be available in mid-August. This data will be compared to the retention data from previous years. We are also looking at the performance of the students in their mathematics and science courses as compared to previous years. In addition, these students will being going out on their first co-op this coming year. We plan to look at the co-op employer survey data and compare student performance on their first co-op job to the baseline data from previous years. The co-op employer survey has several questions pertaining to team-work, communication skills, and problem solving ability.

This year, we plan to implement a flipped pedagogy in Engineering Models I and II. We will compare student performance on exams, projects, and assignments with performance from last year. We are also adding some combination software/hardware experiments to Models I and II and hope to motivate some of the students who view their engineering disciplines as "non-computing" disciplines. The end of course survey will be modified to include specific questions about the team projects.

ACKNOWLEDGMENT

The authors would like to acknowledge the students whose projects were highlighted in this paper: Battleship (Aaron Cunningham, Henry Jentz, Kendrick Li) Adventure (Chelsea Duran, Michael Strohofer, Aaron Trachtenburg), Puzzle Game (Jacob Hall, Josh Myers, Ryan Vanderhart), Organic Chemistry (Colin Dorey, Bryan Langlois, Mack McNamee), Circuits (Kyle Abner, John Cooker, Brian Tsen), ALGO-RACE (Kevin Ernst, Allyssa Griffith, Robert Ipach), and Integral Concepts (Christopher Groh, Adam Kozerski, John Miller, Joshua Quach).

REFERENCES

[1] J. Froyd and M. Ohland, "Integrated Engineering Curricula," Journal of Engineering Education, 147-164, (2005).

[2] L. Cui, N.S. Rebello, and A.G. Bennett, "College Students' Transfer from Calculus to Physics," AIP Conference Proceedings 818, 37-40, (2005).

[3] P. Heller, R. Keith, and S. Anderson, "Teaching Problem Solving Through Cooperative Grouping. Part 1: Group Versus Individual Problem Solving," Am. J. Phys. 60, 627-636 (1992).

[4] D. R. Woods, A. N. Hrymak, R. R. Marshall, P. E. Wood, C. M. Crowe, T. W. Hoffman, J. D. Wright, P. A. Taylor, K. A. Woodhouse, and C. G. K. Bouchard, "Developing Problem Solving Skills: The McMaster Problem Solving Program," ASEE J. of Engr. Educ. 86, 75-91, (1997).

[5] F. P. Deek, H. Kimmel, J. A. McHugh. "Pedagogical changes in the delivery of the first-course in computer science: problem solving, then programming." Journal of Engineering Education 87, 3, 313-320 (1998).

[6] National Academy of Engineering. "The engineer of 2020 : visions of engineering in the new century." Washington, DC, National Academies Press, 2004.

[7] C. L. Dym, "Learning engineering: design, languages, and experiences." Journal of Engineering Education 88, 2, 145-148, (1999).

[8] Pendergrass, N, A, Kowalczyk, R, E, Dowd, J, P, Laoulache, R, N, Nelles, W, et al, "Improving first-year engineering education", *Frontiers in Education*, (1999).

[9] P. N. Johnson-Laird, "Mental models: towards a cognitive science of language, inference, and consciousness." Cambridge, MA, Harvard University Press, 1983.

[10] J. D. Bransford, A. L. Brown, R.R. Cocking, Eds. "How people learn: brain, mind, experience, and school." Washington, D.C., National Academies Press, 1999.

[11] L.L. Constantine, "Constantine on peopleware." Englewood Cliffs, NJ, Yourdon Press, 1995.

[12] C. McDowell, L. Werner, H. Bullock, J. Fernald, "The effects of pair-programming on performance in an introductory programming course." SIGCSE Bull. 34, 1, 38-42 (2002).

[13] C. McDowell, L. Werner, H. Bullock, J. Fernald, "The impact of pair programming on student performance, perception and persistence." International Conference on Software Engineering, Portland, Oregon (2003).

[14] C. A. Bagley, C. C. Chou, "Collaboration and the importance for novices in learning Java computer programming." SIGCSE Bull. 39, 2, 211-215 (2007).

AUTHOR INFORMATION

Kathleen Ossman Associate Professor, Department of Engineering Education, University of Cincinnati, kathy.ossman@uc.edu

Gregory Bucks Assistant Professor - Educator, Department of Engineering Education, University of Cincinnati, gregory.bucks@uc.edu