An Active Learning Module to Introduce Students to the Importance of Flowcharts and Technical Documentation

Mark M. BUDNIK, Rebecca THOMAS, Stewart THOMAS
Electrical and Computer Engineering, Valparaiso University
Valparaiso, IN USA

and

Nicholas ROSASCO
Computing and Information Sciences, Valparaiso University
Valparaiso, IN USA

ABSTRACT

The nature of problems typically tackled in initial programming courses can lead students to dismiss problem solving techniques and processes. Faculty and students in Programming 1 and 2 classes are prone to focus on the mechanics and syntax of programming at the expense of building increasingly complex and realistic models of the solution. To ***motivate*** undergraduate students to accept the importance of flowcharts, pseudo-code, documentation, and other development tools, we present a series of active learning lessons for introductory programming classes built and tested at Valparaiso University. These lessons introduce development tools and problem solving as vital components of an overall solution and provide opportunities for students to see their value in real world applications. Following the lessons, 100% of the students recognized the importance of development tools, and 95% of the students identified a desire to learn more about how they can be used.

Keywords: CS1, Programming 1, Flowcharts, Technical Documentation, Development Tools.

1. INTRODUCTION

Introductory programming classes are often overloaded to maximize the amount of programming content [1] and lack sufficient attention to the organization of computer programs and algorithms through the use of flowcharts and other technical documentation. When included, students perceive these topics as unimportant, trivial, or rather obvious and brush them aside rather than identifying essential skills to add to their growing repertoire.

To motivate undergraduate students to embrace the idea that flowcharts and technical documentation are important, this paper presents a series of active learning lessons suitable for use in Programming 1 and Programming 2 classes. The lessons introduce these topics as vital components to an overall solution and provide opportunities for students to perceive their value in real world applications.

Section 2 provides an introduction of our three active learning lessons. Sections 3 and 4 provide a summary of our assessment and evaluation of the lessons. Finally, Section 5 presents our conclusions.

2. ACTIVE LEARNING LESSONS

Valparaiso University has developed and tested three active learning modules to help demonstrate the importance of flowcharts and technical documentation in the development of programs, algorithms, and systems. The lessons are structured to allow instructors to provide a significant amount of assistance at the start of the first lesson, but reduce the levels of assistance as students work through the three lessons. The duration and technical difficulty of each subsequent activity increases. Each lesson includes manipulatives to address different learning styles. The modules are suitable for implementation in a typical 2 – 3 hour laboratory session, or they can be implemented individually during traditional 50 minute lecture periods [2].

While it is appropriate to use the lessons as separate modules, we carried out all three lessons in sequence with the same group of students. The first and second lessons followed the same basic procedure (see Table 1). Students formed into teams of 2 – 4 people. Each team was presented with a toy puzzle and given the task of **describing** the fastest way to complete the puzzle. Groups were given time to understand how the toy operates and write directions (Steps 1 and 2). After collecting the instruction sets, a faculty member was given a brief period to review (Step 3) and then execute (Step 4) the student directions without assistance from the students. These trials were performed with the students as the audience where the quality metric was the time needed for the faculty member to execute the directions. Next, students considered and discussed the various solutions during a gallery walk (Step 5) [3, 4].

To conclude the lessons, the faculty member summarized the techniques that worked and those that didn't and asked students if it was frustrating that they weren't allowed to talk during the execution of their written directions (Step 6). Best practices and general challenges in both problem solving and communicating operational instructions were then discussed.

Table 1. Experimental Sequence for Lessons 1 and 2.

Step	Lesson 1	Lesson 2	Content
1	3 minutes	N / A	Understand toy operation
2	4 minutes	15 minutes	Write directions
3	1 minute	1–2 minutes	Faculty review
4	(varied)	(varied)	Execution
5	5 minutes	5 minutes	Gallery walk
6	3 minutes	3 minutes	Roundup discussion

Lesson 1: Tupperware Shape-O Toy
In the first lesson, each team was given a Tupperware Shape-O Toy (see Figure 1) [5]. The toy is a dodecahedron shape with ten different holes (for example, cross, square, triangle, circle, etc.) and includes ten different, numbered, yellow plastic shapes that each fit through only one of the ten holes. Two of the dodecahedron sides are used for handles to carry and open the toy. The teams were instructed to determine and *document* the fastest way to put all ten shapes into the respective holes of the Shape-O Toy.

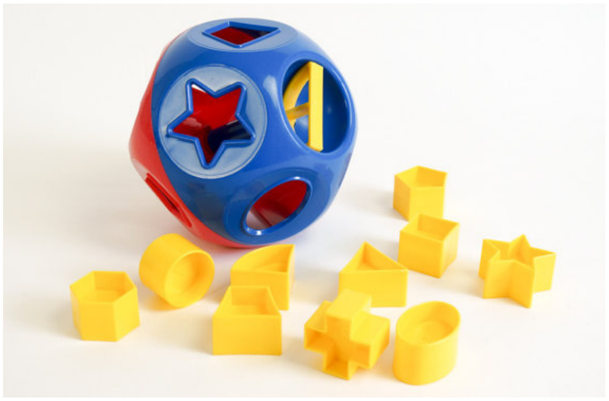


Figure 1. Tupperware Shape-O Toy consisting of a dodecahedron with ten unique shapes. Each shape fits into a hole on only one face of the dodecahedron.

Lesson 2: Superfection
In the second lesson, students remained in the same teams. Students gathered around a central table and the faculty member introduced the Superfection game shown in Figure 2. Superfection [6] is an extended version of the more common Perfection [7] game.

Superfection consists of a base unit (which includes a timer with a start/stop switch) and 32 colored plastic puzzle pieces. Each of the 32 pieces is unique and can be matched with exactly one other piece to form a cube. The object of the game is to match the 32 colored plastic pieces into 16 cubes and place them on the board in as little time as possible. (Note, the 16 cubes can be positioned in any order on the base unit. Their exact position on the base unit is unimportant.)

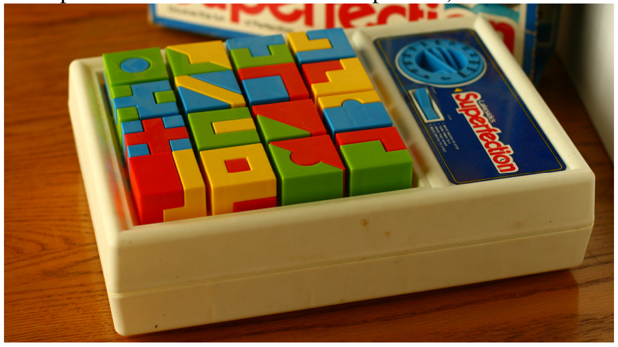


Figure 2. The Superfection game has 32 plastic puzzle pieces that each have precisely one matching puzzle piece to form one of 16 cubes.

Teams each took one Superfection game and were given the task of creating written instructions to assemble and place the 32 Superfection game pieces as quickly as possible. During this step, students were instructed that each team could only ask the faculty member two questions and also there would be a prize awarded for this round (based upon how well their instructions worked).

As with the previous lesson, after the students were finished, faculty executed the directions, students performed a gallery walk, and a concluding discussion was held (see Table 1). The best performing team was determined based on the faculty's fastest instruction execution time.

Lesson 3: Theme Park Attraction
In the final lesson, students saw the importance of efficient algorithm design and practiced using development tools with a simulated theme park attraction control system. This provided students the opportunity to develop a process flow for a real world application and see how their work impacts solution efficiency.

Students were told they are engineers and programmers at a theme park. Unfortunately, a previous team responsible for developing an attraction did not do a good job of documenting their work. The students were tasked with developing the flowchart and documentation for operating the theme park attraction in an efficient manner.

The students were then split into two rooms. The Tower of Terror attraction [8] at Disney Hollywood Studios was used for one room of students. The other room of students used the Grizzly River Run attraction [9] at Disney California Adventure. Faculty members in each room explained to the students how their attractions worked [10, 11], and gave students a few minutes to ask questions about the attraction.

The students were then given the simulator for their respective attraction [12]. Screen shots from the two simulators are shown in Figures 3 (Tower of Terror) and 4 (Grizzly River Run). The simulators are non-trivial and resemble many industrial automation systems having numerous checkpoints in the flow that affect the critical path. Students must develop an algorithm using inputs from approximately 30 asynchronous sensors and actuate 32 outputs to control the flow of guests and maximize the ride capacity. In our lesson, teams were given 25 minutes to develop their algorithm.

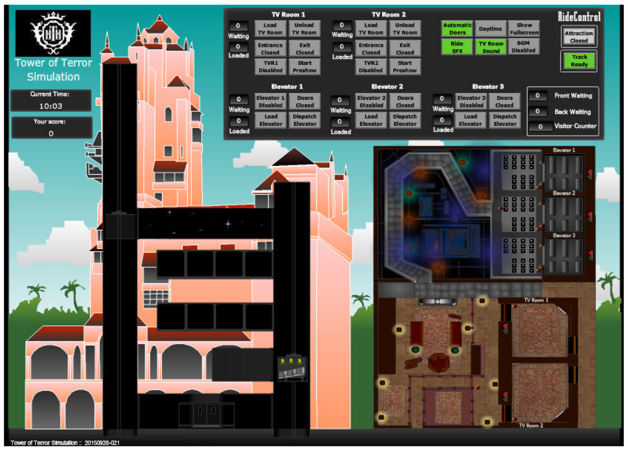


Figure 3. Tower of Terror attraction simulator.

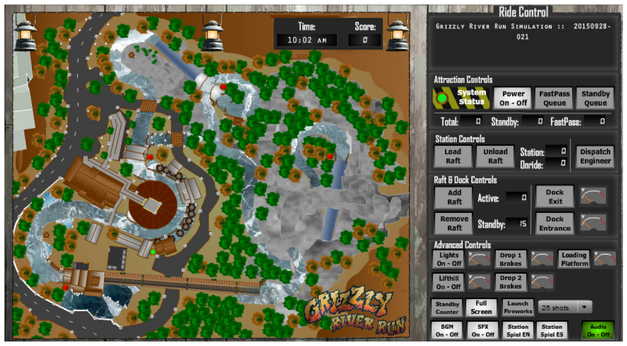


Figure 4. Grizzly River Run attraction simulator.

After the algorithms were complete, the student teams swapped algorithms: Those that worked on the Tower of Terror were given an algorithm for Grizzly River Run, and vice versa. The student teams then had the opportunity to implement the algorithm developed by their peers. They were able to learn how difficult it can be to follow someone else's instructions and the frustration of watching other people try to implement your work.

Finally, the faculty member asked students to summarize what they learned and how their opinions of technical documentation and flowcharting changed in a question and answer session and a post-activity survey.

3. EXTRACURRICULAR ASSESSMENT AND EVALUATION

In this section, we discuss the pre-survey results, provide an analysis of the students' work, present our post-survey results, and provide a brief discussion of the larger learning outcomes in an extracurricular setting.

Pre-Survey Results

Prior to beginning the lessons, students responded to a brief pre-survey consisting of five questions (shown below), each answered on a 5-point, Likert-like scale:

- In my computing assignments, I carefully document how to accomplish a task. (Figure 5)
- I know how to diagram a flow in a standardized way (i.e., UML, pseudocode, flowchart, etc.). (Figure 6)
- I begin tasks by planning my solution before coding the implementation. (Figure 7)
- I believe technical documentation is an important part of the design process. (Figure 8)
- Which is more important to you: Planning a solution or implementing a solution? (Figure 9)

We found that our students generally agree that documentation is an important part of the design process. However, while students believe planning is important, a majority do not routinely plan their work before implementing a software solution. (Responses for the last two questions are shown later in the paper in Figures 8 and 9 alongside student post-survey responses.)

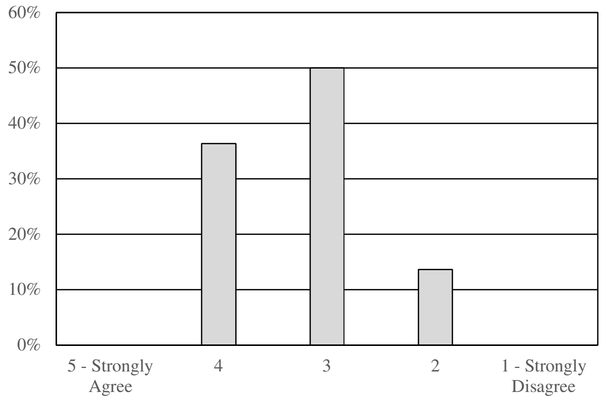


Figure 5. In my computing assignments, I carefully document how to accomplish a task.

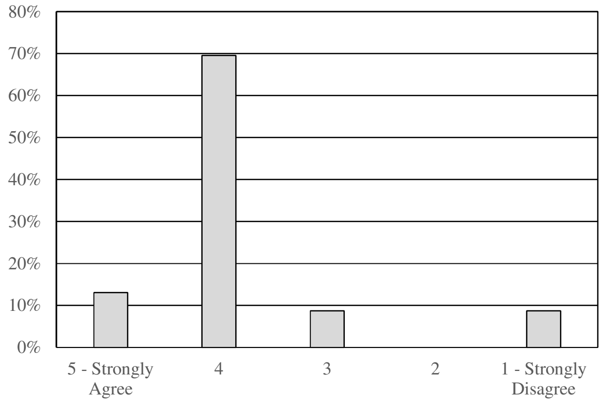


Figure 6: I know how to diagram a flow in a standardized way (i.e., UML, pseudocode, flowchart, etc.).

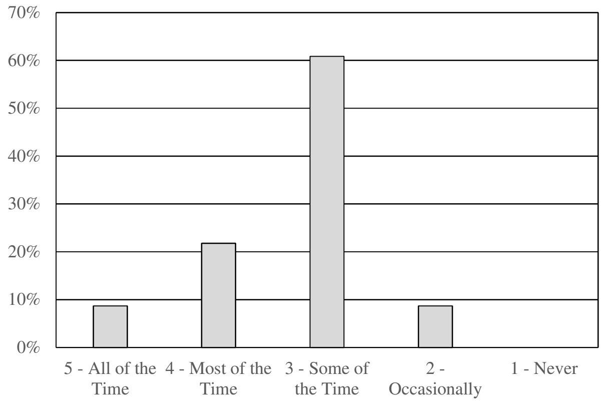


Figure 7: I begin tasks by planning my solution before coding the implementation.

Shape-O Toy

Based on the overall approach, the instructions to solve the Shape-O Toy can be broadly divided into two categories. The instructions with the fastest completion times were variations of: 1) Picking up a shape at random, 2) Rotating the ball to find the corresponding hole, and 3) Continuing until the puzzle was completed.

The second category of instructions were more detailed and provided tasks for each individual piece. For example, one team asked users to find the "cross" piece and put it into the corresponding hole, then find the "circle" piece and put it into the corresponding hole, etc. This resulted in a longer completion time due to the extra steps and careful searching of both the instructions and the pieces involved.

After watching a faculty member follow the instructions and struggle with the Shape-O Toy, the groups gathered to discuss what seemed to work best. Students noted that what seemed good or obvious to them was not evident to someone else reading their instructions. Students were then given an optimized algorithm which allowed a faculty member to insert all ten pieces in a very short time. The optimized algorithm required the user to identify the shape of the hole on the dodecahedron immediately in front of them. Then, they were instructed to visually inspect the ten plastic pieces for the correct shape, insert the correct shape, and then rotate the dodecahedron to the next hole. This process is optimized because it relies upon minimal manipulation of the physical components and takes advantage of the rapid speed at which users can visually identify distinct shapes.

Analogies were then drawn to real world industrial automation systems. Computers and visualization technologies are orders of magnitude faster than physical systems. To optimize system design, sometimes developers must rethink their initial approach to an algorithm to take advantage of the available technologies.

Superfection Game

For the Superfection game, the groups' instructions varied in the amount of detail they attempted to convey and the method used to convey information. Student groups that wrote detailed instructions in list form struggled with describing the non-standard, three-dimensional game pieces. The lack of a shared vocabulary posed a challenge for description and caused confusion during execution. Several groups encountered complications when they used color descriptions of pieces in their instructions and later discovered that the colors of the pieces were not consistent across all game sets.

One of the fastest instructions began by filling each row of the board with a specific color piece that would be the base for a cube. Next, a top piece that would complete a cube was chosen at random and visually matched to a base. Since the 16 base pieces had already been placed, this reduced the sorting problem in half. Additionally, as each cube consists of a different color base and top, the row corresponding to the color of the selected top piece could be eliminated. Another fast instruction set included a top-level drawing of the completed solution.

Other instructions that took longer to solve used more detailed descriptions and matched a top and bottom to make the cube before placing them on the board. This seemed to work out well when small groups of game pieces could be identified by features such as a rounded edge or connection on a diagonal, but failed when it was more difficult to quickly describe commonalities between complex game pieces.

After performing the gallery stroll, students thought that using a diagram of the solution was a best practice and noted that it was helpful to see what you are trying to make. Additionally, students recognized that finding similarities between pieces and trying to identify them was a strategy that did not work well.

Theme Park Attraction Simulators

Students were given the most time to work on this lesson, and their work showcased a significant amount of effort to first understand how the relatively complex attraction controls worked and then effectively document their operation.

Dividing the student teams into pairs resulted in two positive effects. First, with pairs, the groups developing the documentation were smaller and each member was fully involved in the process. Second, unbeknownst to each other, the pairs were working on separate projects for their peers to implement. Therefore, each pair got the opportunity to not only develop a flowchart and process for their teammates, but also to implement their classmates' work.

The teams that did best on the third lesson tended to remain the most focused on the task across the entire 25-minute span. In retrospect, it would have been better to allow students more time to complete this lesson. By allowing up to 40 minutes, the students would have the opportunity to take momentary breaks from the relatively intense time pressure that the simulator presents to the user.

Post-Survey Results

After completing the three lessons, students were given a post-survey consisting of the following five questions scored on a 5-point Likert-like scale:

- I believe technical documentation is an important part of the design process. (Figure 8, also in pre-survey)
- Which is more important to you: Planning a solution or implementing a solution? (Figure 9, also in pre-survey)
- I would like to know more about how to diagram a flow in a standardized way (i.e., UML, pseudocode, flowchart, etc.). (Figure 10)
- This workshop helped me understand the larger challenges of design. (Figure 11)
- I would like to see more activities like this as a part of a regular class. (Figure 12)

The response distributions for the five questions are plotted in Figures 8–12. These responses show that the goal of ***motivating*** students to accept the idea that flow charts and technical documentation are important was successfully met.

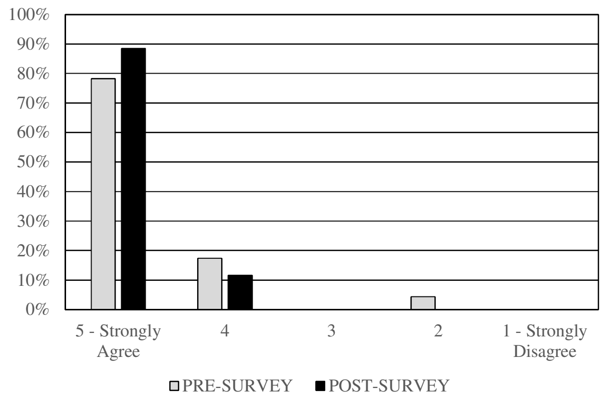


Figure 8. Pre- and post-survey: I believe technical documentation is an important part of the design process.

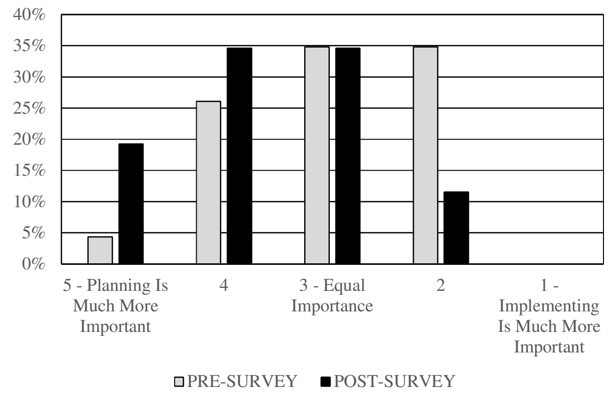


Figure 9. Pre- and post-survey: Which is more important to you: Planning a solution or implementing a solution?

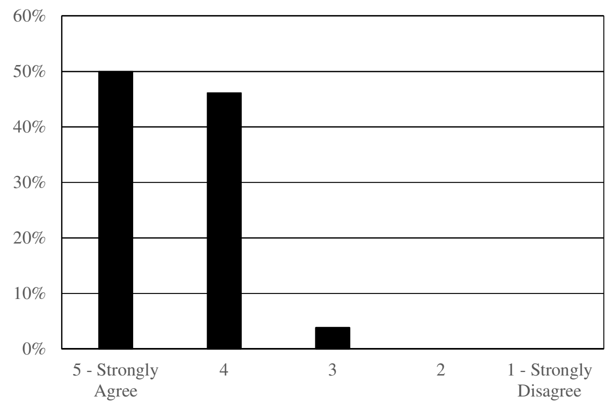


Figure 10. Post-survey only: I would like to know more about how to diagram a flow in a standardized way (i.e., UML, pseudocode, flowchart, etc.).

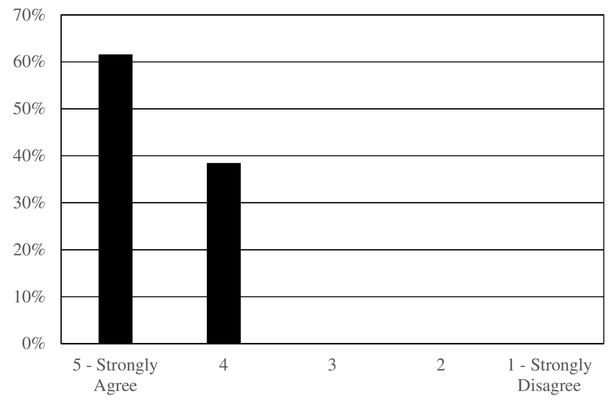


Figure 11. Post-survey only: This workshop helped me understand the larger challenges of design.

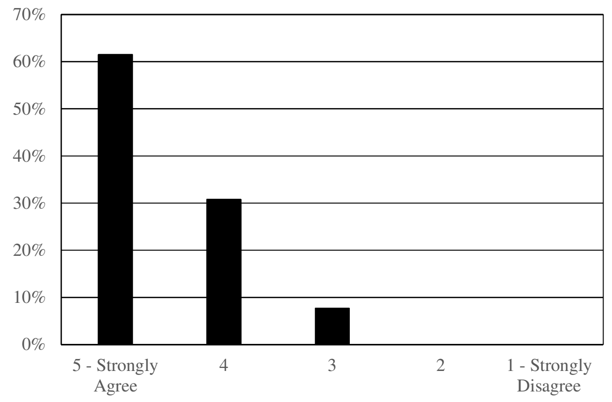


Figure 12. Post-survey only: I would like to see more activities like this as a part of a regular class.

Discussion of Larger Learning Outcomes
The three exercises encompass aspects of various problem spaces such as user interface design and programming languages. They also demonstrate the challenge in technical writing for an unknown audience. The certainty of a shared or at least easily grasped vocabulary eases the challenge of creating documentation for the Shape-O Toy. For the Superfection game and attraction simulators, the inability to use a simple vocabulary significantly raised the complexity challenge and demonstrated the increased likelihood of end-user confusion. These lessons demonstrate the risks that can occur when directions cannot be easily understood. Such failures to communicate clearly are frequent contributors to accidents, mishaps, and dangerous misuse. The use of color-based descriptions also provides the opportunity to discuss accessibility issues as important considerations in the design phase.

The differences in task complexity across all three exercises can be used to discuss operational problems in the context of complete systems. The whole exercise and the contrasts among the lessons serves as a way to introduce the vocabulary of processes and algorithms. Asking students to identify measures and metrics and define success could be a follow-up activity or out of class assignment. Further discussion could also include questions of labeling steps for repetition (looping), how to handle questions (conditionals), and initial conditions (initialization).

Additionally, we see value in these exercises beyond introductory programming courses. For example, the complexity of Superfection versus the simpler Shape-O Toy could be illustrate the comparison of CISC versus RISC devices in a computer architecture course. These exercises are also compelling for project-driven or capstone-style courses with a strong technical writing or documentation component.

4. CLASSROM ASSESSMENT AND EVALUATION

In this section, we discuss how the various modules were integrated in parts in two separate classes. The Tupperware Shape-O Toy and Superfection game were used in a sophomore level Programming 1 course (ECE251) for electrical and computer engineering students. The theme park attraction control simulators were used in a general engineering design elective (ECE490) populated primarily by sophomores and juniors and included students majoring in biomedical, civil, computer, electrical, and mechanical engineering.

ECE251 / Programming 1 Implementation

ECE251 is an introductory programming course designed for second year electrical engineering and computer engineering students. For many students, this course is an initial exposure to computer programming. ECE251 uses the C programming language to teach students the basics of memory, variable usage, program compilation and structured programming using functions. Upon completion, students are prepared for a second semester of object-oriented programming or a course in embedded microcontroller program design.

As an introductory computer programming course, it is important for students to understand the basics of program design and need for careful planning. To motivate students to see the need for tools like flowcharts and pseudocode, a subset of the modules presented in this paper were administered to the students during a regular course meeting.

The structure for this activity is shown in the table below. This lesson was administered by a single professor and one undergraduate student in two sections of ECE251. While each section meeting is 75 minutes, the activity was planned to last for approximately 50 minutes with a short lecture taking up the meeting balance. Students were divided into teams of two for this activity.

Table 2. Experimental Sequence for ECE251.

Item	Time (min.)
Administrative items • Daily quiz • Announcements	5
Lecture on new material	15
Shape-O Module • Understand toy, determine fastest solution • Write instructions	10-15
Professor and TA implement Shape-O instructions • Discussion of algorithm differences	10
Superfection Module • Understand toy, determine fastest solution • Write instructions	10-15
Discussion of Shape-O and Superperfection instructions	5
Overall discussion on documentation	5-10
TOTAL TIME	60-75

This activity took place three weeks into a 15 week semester. Students were capable of writing basic programs (i.e., calculate the area of a triangle and print the result) but had not yet seen a need for a structured design tool. Along with this activity, students were given a homework assignment to produce a flowchart that outlines their process for printing the binary representation of a 32-bit number stored in memory. This was the students' first assignment that asked for a flowchart (see Figures 12 and 13).

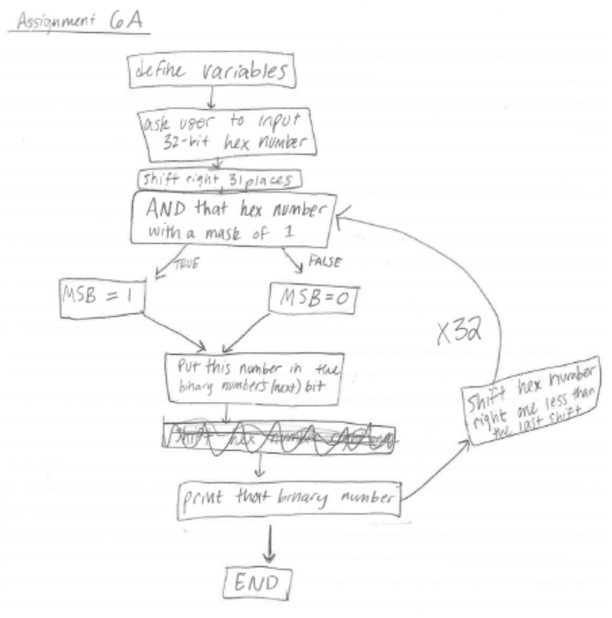

Assignment 6A
define variables
ask user to input 32-bit hex number
shift right 31 places
AND that hex number with a mask of 1
TRUE
FALSE
MSB = 1
MSB=0
Put this number in the binary numbers (next) bit
~~Shift hex number~~
print that binary number
END
X32
Shift hex number right one less than the last shift

Figure 12. Example of ECE251 algorithm for printing a binary representation of a 32-bit number that meets expectations.

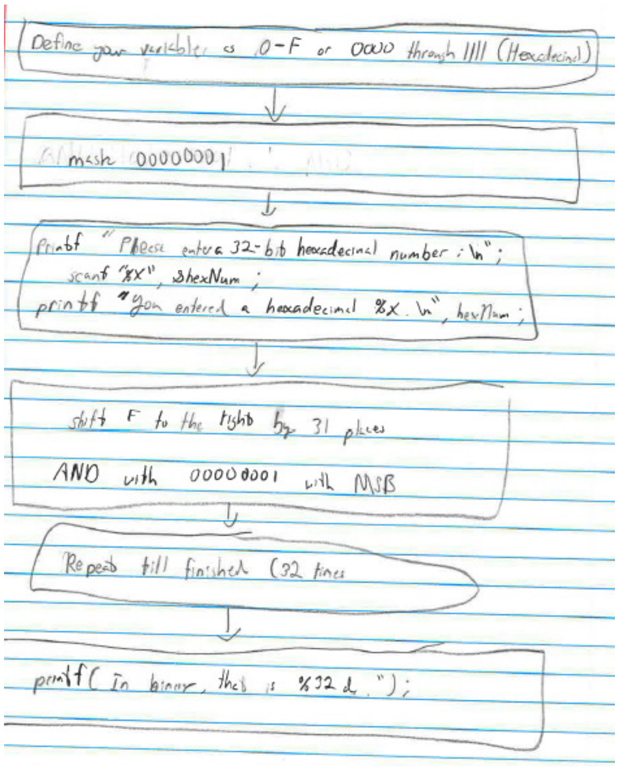

Define your variables as 0-F or 0000 through 1111 (Hexadecimal)
mask 00000001
Printf " Please enter a 32-bit hexadecimal number : \n";
scanf "%X", &hexNum ;
printf "you entered a hexadecimal %X. \n", hexNum ;
shift F to the right by 31 places
AND with 00000001 with MSB
Repeat till finished (32 times
printf(In binary, that is %32 d. ");

Figure 13. Example of ECE251 algorithm for printing a binary representation of a 32-bit number that is progressing toward expectations.

The impact of this activity was assessed in two ways. Student responses to surveys administered before and after the activity were assessed (see Figure 14). Second, student performance on their flowchart assignment was assessed by the professor (see Figure 15).

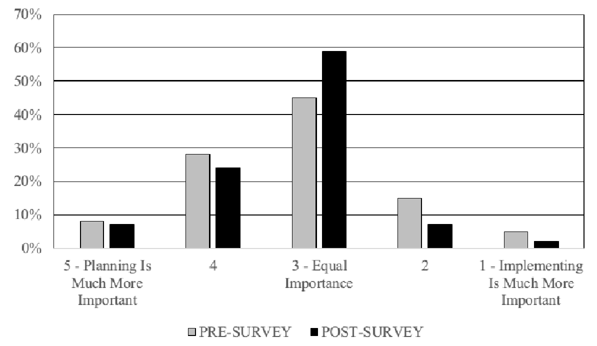


Figure 14. Pre- and post-survey ECE251: Which is more important to you: Planning a solution or implementing a solution?

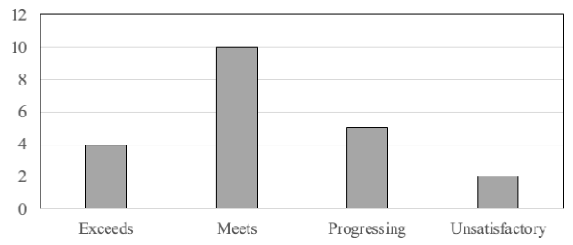


Figure 15. Assessment of ECE251 flowchart assignment.

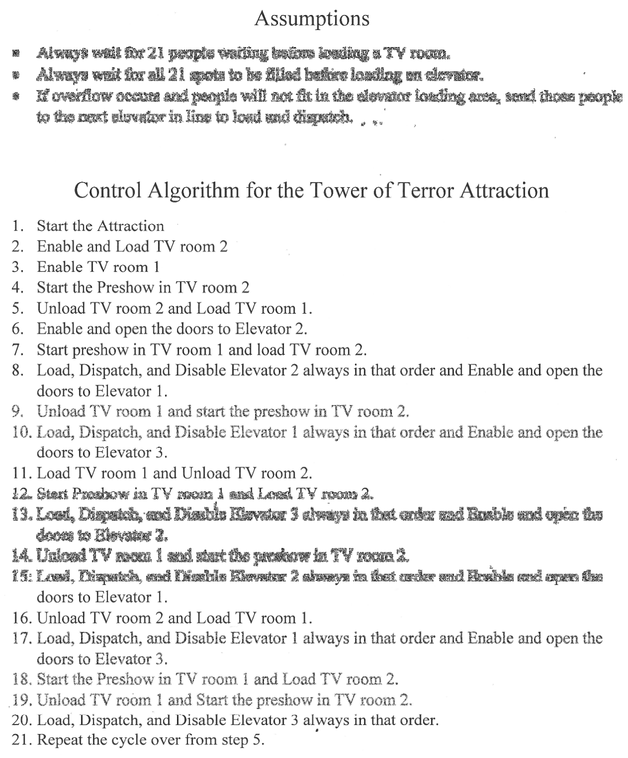
Data from the assessment show that students recognized the need for planning complex tasks. Further, data from the flowchart assignment show that a large set of students were capable of creating documentation for a programming task with little previous guidance. This creates a strong starting point for continuing to practice design and documentation procedures regardless of specific discipline.

This activity shows one example of how the modules presented in this paper function in a typical classroom setting. Data from the ECE251 course suggests that the modules are strongest when taken together. In this offering, students only saw a few examples of a professor implementing the Shape-O instructions and no examples of the Superperfection instructions. To make a strong impact, it is suggested to run a short 20 minute activity with the Shape-O toy on one day, a 20 minute activity with Superperfection the next day, and then culminate on day 3 with the theme park attraction simulators.

ECE490 / Innovation in Engineering Design
ECE490 is an engineering elective for all engineering majors. It provides additional opportunities for students to engage in design by providing multiple short-term projects each focused on a specific step in the design process or various design tools.

As outlined in Section 2, Lesson 3, ECE490 students were assigned the role of engineers at a theme park with the task of developing the documentation for operating an attraction in an efficient manner. Again, the task is non-trivial as they must develop an algorithm using inputs from approximately 30 asynchronous sensors and actuate 32 outputs to control the flow of guests and maximize the ride capacity. In this lesson, however, the ECE490 students were given a week to learn how the simulator works and develop their flowchart. Also, unlike the extracurricular implementation, a significant portion of the ECE490 students (biomedical, civil, and mechanical engineering students) have not had a Programming 1 course, and did not have a prior introduction to flowcharting. The electrical and computer engineering students in ECE490 had completed a Programming 1 course, but had not yet been introduced to interrupt driven systems.

The students performed admirably on the exercise, despite the majority lacking a strong background in flow charting or related processes. The additional time allowed the students to truly understand the intricacies of the simulator's operation and develop, generally, very robust control algorithms. They ranged from pseudocode descriptions (see Figure 16) to more detailed flow charts (see Figure 17), all the way to recognizing the need for an interrupt driven system (see Figure 18). In all, 27% of the student work was classified as Progressing Toward Expectations, and 73% of the submissions were deemed to Meet or Exceed our expectations. Additionally, the investment by the student teams in the assignment allowed for a much richer understanding of the control systems and provided a canvas for a much deeper, student-led discussion of optimizing a system's flow.



Assumptions

- Always wait for 21 people waiting before loading a TV room.
- Always wait for all 21 spots to be filled before loading an elevator.
- If overflow occurs and people will not fit in the elevator loading area, send those people to the next elevator in line to load and dispatch.

Control Algorithm for the Tower of Terror Attraction

```
1. Start the Attraction
2. Enable and Load TV room 2
3. Enable TV room 1
4. Start the Preshow in TV room 2
5. Unload TV room 2 and Load TV room 1.
6. Enable and open the doors to Elevator 2.
7. Start preshow in TV room 1 and load TV room 2.
8. Load, Dispatch, and Disable Elevator 2 always in that order and Enable and open the
   doors to Elevator 1.
9. Unload TV room 1 and start the preshow in TV room 2.
10. Load, Dispatch, and Disable Elevator 1 always in that order and Enable and open the
   doors to Elevator 3.
11. Load TV room 1 and Unload TV room 2.
12. Start Preshow in TV room 1 and Load TV room 2.
13. Load, Dispatch, and Disable Elevator 3 always in that order and Enable and open the
   doors to Elevator 2.
14. Unload TV room 1 and start the preshow in TV room 2.
15: Load, Dispatch, and Disable Elevator 2 always in that order and Enable and open the
   doors to Elevator 1.
16. Unload TV room 2 and Load TV room 1.
17. Load, Dispatch, and Disable Elevator 1 always in that order and Enable and open the
   doors to Elevator 3.
18. Start the Preshow in TV room 1 and Load TV room 2.
19. Unload TV room 1 and Start the preshow in TV room 2.
20. Load, Dispatch, and Disable Elevator 3 always in that order.
21. Repeat the cycle over from step 5.
```

Figure 16. Pseudocode description of student control algorithm for Tower of Terror theme park attraction.

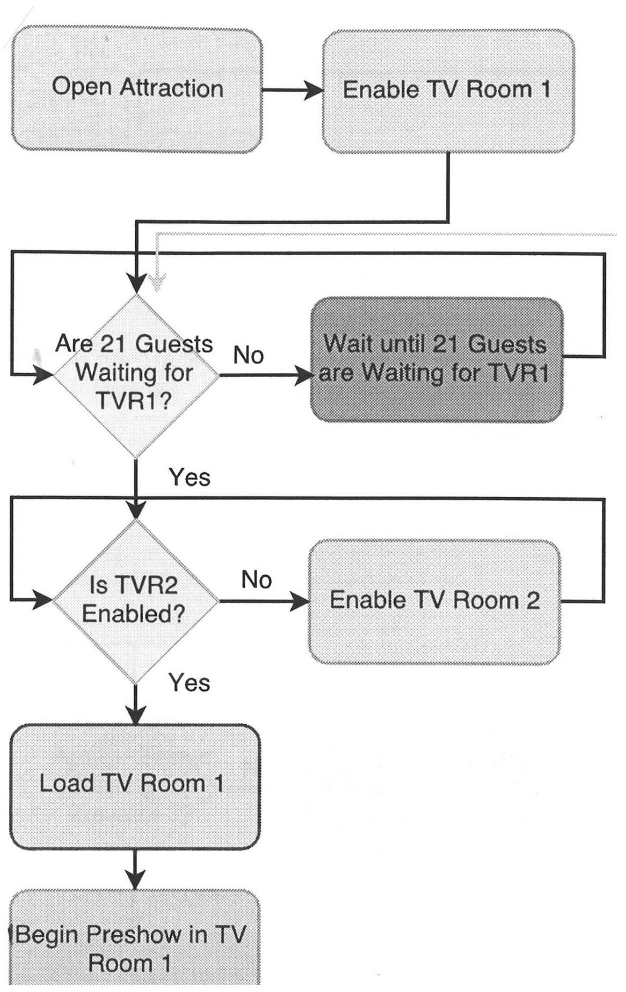

Open Attraction
Enable TV Room 1
Are 21 Guests Waiting for TVR1?
No
Wait until 21 Guests are Waiting for TVR1
Yes
Is TVR2 Enabled?
No
Enable TV Room 2
Yes
Load TV Room 1
Begin Preshow in TV Room 1

Figure 17. Excerpt of detailed flowchart of student control algorithm for Tower of Terror theme park attraction.

From an educator's perspective, the most refreshing aspect of the inclusion of our lessons in the classes was that the students "got it." They able to demonstrate reasonable work toward developing system documentation and flowcharts. Each group showed progress toward flowcharting maturity, and showed improvement in motivation and understanding of flowcharting and planning. Even allowing for a subset of the three lessons in ECE251 and ECE490, each variation showed improvement even with the less immersed experience.

Additionally, each group had some sort of outcome that was unexpected. In ECE251, we had a "hook" that opened the door for the instructor to start a discussion of pseudocode and other planning tools. In ECE490, submissions allowed us to introduce the concept of functions and subroutines for the repetition of activities and thinking about synchronous vs. asynchronous behavior. Finally, following the submission of the student work, the documentation provided material for a thorough, thoughtful conversation into best practices and areas for improvement.

5. CONCLUSION

We have presented a series of active learning lessons to show development tools and problem solving as vital components of an overall solution. These lessons ***motivate*** students to see the value of such tools through immediate, practical applications. Assessment of pre- and post-lesson surveys indicate that our students, after participating in these exercises, are not only motivated to use suitable tools and planning in their coding projects, but are also interested in learning more about standard documentation and planning tools. In future work, we will continue refining our modules and further study their role within the curricula.

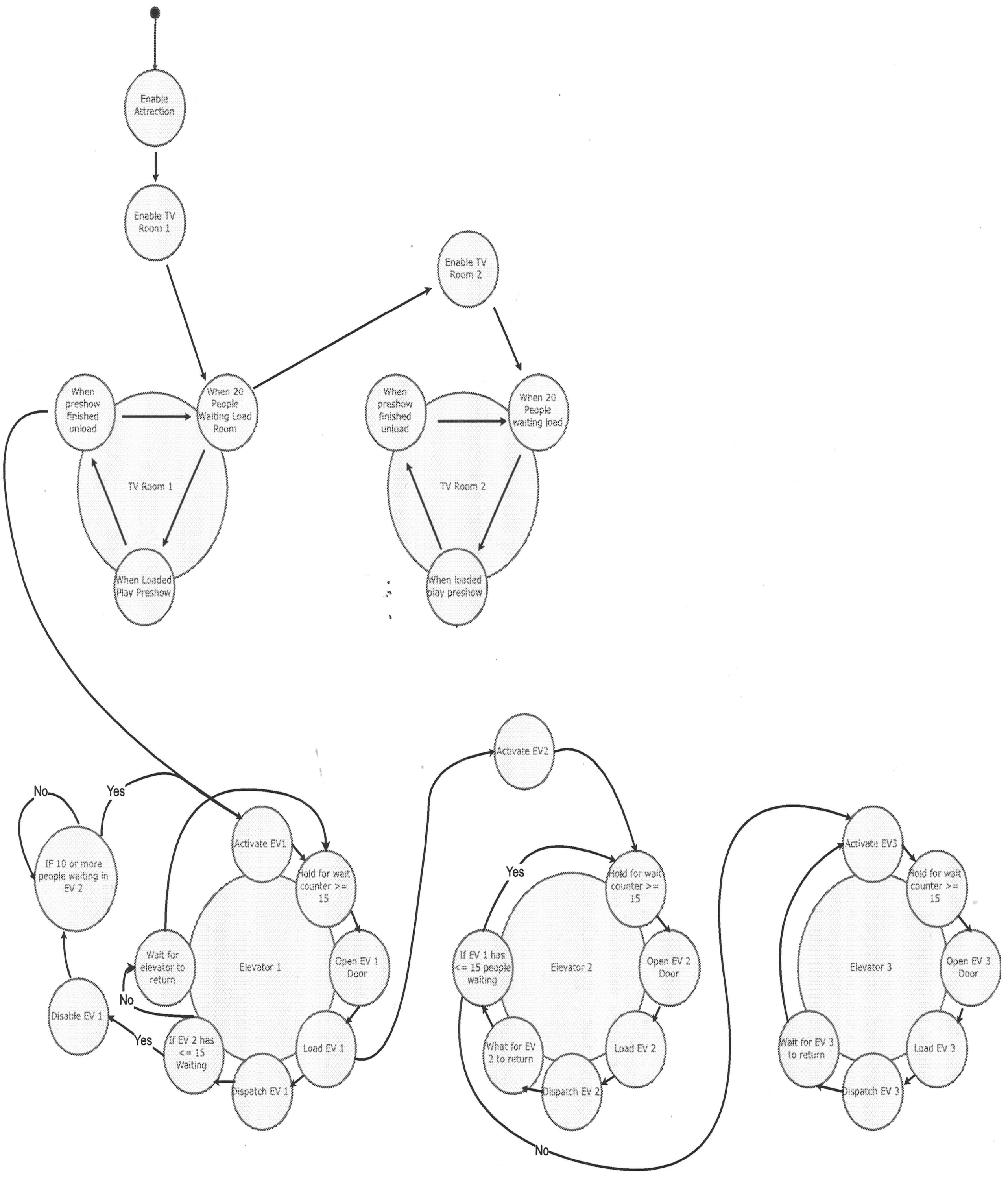


Figure 18. Detailed flowchart of interrupt driven student control algorithm for Tower of Terror theme park attraction.

6. REFERENCES

[1] D.Baldwin, V.Barr, A.Briggs, J.Havill, B.Maxwell, H.M.Walker, "CS 1: Beyond Programming," Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, Seattle, WA, March 8-11, 2017.

[2] M.M.Budnik, R.Thomas, N.Rosasco, S.Thomas, "An Active Learning Module to Introduce Students to the Importance of Flowcharts and Technical Documentation," Proceedings of the 2017 EISTA Conference on Educaiton and Information Systems, Technologies, and Applications, Orlando, FL, July 8-11, 2017.

[3] J. L.Kolonder, "Facilitating the Learning of Design Practices: Lesson Learned from and Inquiry into Science Education," *Journal of Industrial Teacher Education*, 39(3), 2002, pp. 1-31.

[4] B.Fasse, and J.L.Kolodner, "Evaluating Classroom Practices Using Qualitative Research Methods: Defining and Refining the Process," Fourth International Conference of the Learning Sciences, pp. 193-198, Mahwah, NJ: Erlbaum.

[5] Tupperware Brands Corporation, "Shape-O® Toy," http://www.tupperware.com/shape-o-toy-4657.html, retrieved March 31, 2017.

[6] Lakeside Games, "Superfection," boardgamegeek.com/boardgame/8943/superfection, retrieved March 31, 2017.

[7] Hasbro Games, "Perfection," www.hasbro.com/en-us/product/the-original-game-of-perfection:8F011721-6D40-1014-8BF0-9EFBF894F9D4, Retrieved March 31, 2017.

[8] Walt Disney World, "The Twilight Zone Tower of Terror™, disneyworld.disney.go.com/attractions /hollywood-studios/twilight-zone-tower-of-terror/, Retrieved March 31, 2017.

[9] Disney California Adventure, "Grizzly River Run," https://disneyland.disney.go.com/attractions/disney-california-adventure/grizzly-river-run/, retrieved March 31, 2017.

[10] M.Smith, "Twilight Zone Tower of Terror," http://www.martinsvids.net/?p=626, retrieved March 31, 2017.

[11] Tower Secrets, "Inside the Tower of Terror Construction," http://towersecrets.com/tower-of-terror-construction-video-translation/, Retrieved March 31, 2017.

[12] Themagical, "Control Your Favorite Ride," https://www.themagical.nl/, retrieved March 31, 2017.