# The Journal of Computing Sciences in Colleges

## Papers of the 26th Annual CCSC Northeastern Conference

April 16th-17th, 2021
Ramapo College of New Jersey
Mahwah, NJ

Baochuan Lu, Editor
Southwest Baptist University

Jeremiah W. Johnson, Regional Editor
University of New Hampshire

**Volume 36, Number 8** **April 2021**

*The Journal of Computing Sciences in Colleges* (ISSN 1937-4771 print, 1937-4763 digital) is published at least six times per year and constitutes the refereed papers of regional conferences sponsored by the Consortium for Computing Sciences in Colleges.

# Table of Contents

## The Consortium for Computing Sciences in Colleges Board of Directors

Following is a listing of the contact information for the members of the Board of Directors and the Officers of the Consortium for Computing Sciences in Colleges (along with the years of expiration of their terms), as well as members serving CCSC:

**Karina Assiter**, President (2022), (802)387-7112, karinaassiter@landmark.edu.

**Chris Healy**, Vice President (2022), chris.healy@furman.edu, Computer Science Department, 3300 Poinsett Highway Greenville, SC 29613.

**Baochuan Lu**, Publications Chair (2021), (417)328-1676, blu@sbuniv.edu, Southwest Baptist University - Department of Computer and Information Sciences, 1600 University Ave., Bolivar, MO 65613.

**Brian Hare**, Treasurer (2020), (816)235-2362, hareb@umkc.edu, University of Missouri-Kansas City, School of Computing & Engineering, 450E Flarsheim Hall, 5110 Rockhill Rd., Kansas City MO 64110.

**Cathy Bareiss**, Membership Secretary (2022), cathy.bareiss@betheluniversity.edu, Department of Mathematical Engineering Sciences, 1001 Bethel Circle, Mishawaka, IN 46545.

**Judy Mullins**, Central Plains Representative (2023), Associate Treasurer, (816)390-4386, mullinsj@umkc.edu, UMKC, Retired.

**Michael Flinn**, Eastern Representative (2023), mflinn@frostburg.edu, Department of Computer Science Information Technologies, Frostburg State University, 101 Braddock Road, Frostburg, MD 21532.

**David R. Naugler**, Midsouth Representative(2022), (317) 456-2125, dnaugler@semo.edu, 5293 Green Hills Drive, Brownsburg IN 46112.

**Grace Mirsky**, Midwest Representative(2023), gmirsky@ben.edu, Mathematical and Computational Sciences, 5700 College Rd. Lisle, IL 60532.

**Lawrence D'Antonio**, Northeastern Representative (2022), (201)684-7714, ldant@ramapo.edu, Computer Science Department, Ramapo College of New Jersey, Mahwah, NJ 07430.

**Shereen Khoja**, Northwestern Representative(2021), shereen@pacificu.edu, Computer Science, 2043 College Way, Forest Grove, OR 97116.

**Mohamed Lotfy**, Rocky Mountain Representative (2022), Information Systems & Technology Department, College of Engineering & Technology, Utah Valley University, Orem, UT 84058.

**Tina Johnson**, South Central Representative (2021), (940)397-6201, tina.johnson@mwsu.edu, Dept. of Computer Science, Midwestern State University, 3410 Taft Boulevard, Wichita Falls, TX 76308.

**Kevin Treu**, Southeastern Representative (2021), (864)294-3220, kevin.treu@furman.edu, Furman University, Dept of Computer Science, Greenville, SC 29613.

**Bryan Dixon**, Southwestern Representative (2023), (530)898-4864,

bcdixon@csuchico.edu, Computer Science Department, California State University, Chico, Chico, CA 95929-0410.

**Serving the CCSC:** These members are serving in positions as indicated:

**Bin Peng**, Associate Editor, (816) 584-6884, bin.peng@park.edu, Park University - Department of Computer Science and Information Systems, 8700 NW River Park Drive, Parkville, MO 64152.

**Shereen Khoja**, Comptroller, (503)352-2008, shereen@pacificu.edu, MSC 2615, Pacific University, Forest Grove, OR 97116.

**Elizabeth Adams**, National Partners Chair, adamses@jmu.edu, James Madison University, 11520 Lockhart Place, Silver Spring, MD 20902.

**Megan Thomas**, Membership System Administrator, (209)667-3584, mthomas@cs.csustan.edu, Dept. of Computer Science, CSU Stanislaus, One University Circle, Turlock, CA 95382.

**Deborah Hwang**, Webmaster, (812)488-2193, hwang@evansville.edu, Electrical Engr. & Computer Science, University of Evansville, 1800 Lincoln Ave., Evansville, IN 47722.

## CCSC National Partners

The Consortium is very happy to have the following as National Partners. If you have the opportunity please thank them for their support of computing in teaching institutions. As National Partners they are invited to participate in our regional conferences. Visit with their representatives there.

### Platinum Partner

*Turingscraft*
*Google for Education*
*GitHub*
*NSF – National Science Foundation*

### Silver Partners

*zyBooks*

### Bronze Partners

*National Center for Women and Information Technology*
*Teradata*
*Mercury Learning and Information*
*Mercy College*

## Welcome to the 2021 CCSC Northeastern Conference

Welcome to the Twenty Fifth Annual Consortium for Computing Sciences in Colleges Northeast Region Conference. This year's conference is being held virtually. We'd like to thank everyone who has been involved with this conference since its inception at the University of Hartford in April 1996.

This year our program features an outstanding invited speaker, Julia Stoyanovich of New York University. A variety of topics will be covered by the paper presentations, workshops, panels, tutorials, lightning talks, and faculty and student research posters.

A special thanks goes out to the many volunteers who have worked on our conference. This includes the conference committee, the CCSCNE board, and the conference reviewers. You will find their names listed in the committee lists.

The past year has been a rough time for all of us. The 2020 conference had to be cancelled. For this year's conference, out of 12 paper submissions, 7 were accepted for an acceptance rate of 58.3%.

We hope that you enjoy the conference and find it informative and engaging. We look forward to seeing you in 2022 at Pace University in Pleasantville, NY.

Lawrence D'Antonio
Benjamin Fine
Ramapo College of New Jersey
Conference Co-Chairs

## 2021 CCSC Northeastern Conference Steering Committee

Lawrence D'Antonio, Conference Chair ..... Ramapo College of New Jersey
Ben Fine, Conference Chair ..... Ramapo College
Jim Teresco, Program Chair ..... Siena College
Ali Erkan, Papers Chair ..... Ithaca College
Yana Kortsarts, Papers Chair ..... Widener University
Susan Imberman, Lightning Talks Chair ... The City University of New York
Joan DeBello, Panels Chair ..... St. John's University
Bonnie MacKellar, Tutorials and Workshops Chair .... St. John's University
Ting Liu, Tutorials and Workshops Chair ..... Siena College
Dan Rogers, Faculty Posters Chair ..... The College at Brockport
Ingrid Russell, Speakers Chair ..... University of Hartford
Mike Gousie, Speakers Chair ..... Wheaton College (Massachusetts)
Karl Wurst, Student Unconference Chair ..... Worcester State University
Darren Lim, Encore Chair ..... Siena College
Sandeep Mitra, Undergraduate Posters Chair ..... The College at Brockport
Alice Fischer, Undergraduate Posters Chair ..... University of New Haven
Aparna Mahadev, Undergraduate Posters Chair .. Worcester State University
Stefan Christov, Undergraduate Posters Chair ..... Quinnipiac University
Liberty Page, Undergraduate Posters Chair ..... University of New Haven
Mark Hoffman, Registration Chair ..... Quinnipiac University
Rick Kline, Registration Chair ..... Pace University
Frank Ford, Programming Contest ..... Providence College
Del Hart, Programming Contest ..... SUNY Plattsburgh
Scott Frees, Career Fair Coordinator ..... Ramapo College
Kevin McCullen, Vendors Chair ..... SUNY Plattsburgh

## Regional Board — 2021 CCSC Northeastern Region

Lawrence D'Antonio, Board Representative .. Ramapo College of New Jersey
Jeremiah Johnson, Editor ..... University of New Hampshire at Manchester
Mark Hoffman, Registrar ..... Quinnipiac University
Adrian Ionescu, Treasurer ..... Wagner College
Stoney Jackson, Webmaster ..... Western New England University
Ingrid Russell, Secretary ..... University of Hartford

## Reviewers — 2021 CCSC Northeastern Conference

Chris Alvin ..... Furman University
Kailash Chandra ..... Pittsburg
Jami Cotler ..... Siena College
Lawrence D'Antonio ..... Ramapo College
Dan DiTursi ..... Siena College
Martin Gagne ..... Wheaton College
Alessio Gaspar ..... University of South Florida Polytechnic
Michael Gousie ..... Wheaton College (MA)
Scott Harrison ..... St. John Fisher College
Delbert Hart ..... SUNY Plattsburgh
Ahmed Ibrahim ..... University of Pittsburgh
William Joel ..... Graphics Research Group / WCSU
Jeremiah Johnson ..... University of New Hampshire
Zach Kissel ..... Merrimack College
Devorah Kletenik ..... City University of New York
Daniel Krutz ..... Rochester Institute of Technology
Ting Liu ..... Siena college
Sriharsha Mallapuram ..... Plymouth State University
Christopher Martinez ..... University of New Haven
Robert McCloskey ..... University of Scranton
Kevin McCullen ..... SUNY Plattsburgh
Pat Ormond ..... Utah Valley University
Greta Pangborn ..... Saint Michael's College
Sofya Poger ..... Felician University
Christine Reilly ..... Skidmore College
Daniel Rogers ..... The College at Brockport
Tania Roy ..... New College of Florida
Christelle Scharff ..... Pace University
Gurmukh Singh ..... SUNY at Fredonia, NY
Marc Waldman ..... Manhattan College
Yang Wang ..... La Salle University

# Game On: Teaching Cybersecurity to Novices Through the Use of a Serious Game*

*Devorah Kletenik, Alon Butbul, Daniel Chan*
*Deric Kwok, Matthew LaSpina*
*Department of Computer and Information Science*
*Brooklyn College, City University of New York*
*Brooklyn, NY 11210*
*kletenik@sci.brooklyn.cuny.edu*

**Abstract**

We report on the creation of an educational serious game to teach basic cybersecurity concepts. *Cyber Secured* uses engaging gameplay and challenges to educate students about concepts such as phishing, malware, encryption and passwords. This game was evaluated on introductory students in three sections of an e-commerce course. Our observations demonstrated statistically significant learning gains as well as continued retention of the material. We also saw evidence of increased interest in cybersecurity, and reports of positive attitudes towards the use of this game to teach and assess cybersecurity material. The results of our work suggest that *Cyber Secured* is a useful tool to educate about cybersecurity, and we have made our game freely available.

## 1 Introduction

Cybercrime poses a threat to both our society and economy. An increasing awareness of human users as the "weakest link" compels building awareness and educating Internet users about cybersecurity. Many different types of training sessions and exercises have been conducted on a variety of Internet

users. In this work, we discuss our use of a serious game to educate college students who are new to the field of cybersecurity.

Over the past two decades, there has been a growing interest in creating educational "serious games" that help students learn by offering an engaging alternative or supplement to traditional lectures. Research suggests that serious games are superior at teaching subject matter compared to traditional means of instruction and increase long-term retention and student motivation (e.g. [7, 12]). A number of serious games have been created to teach cybersecurity concepts, including digital games, card games and Capture the Flag competitions. These games offer engaging ways to teach about cybersecurity and increase student interest.

However, many of these games are geared towards those who are already knowledgeable about cybersecurity. When novices to the field are overly challenged by cybersecurity games, they may have poor learning outcomes and possibly exhibit counteractive decreased interest in cybersecurity [8]; as a result, it is important to achieve game balance by matching the game topics to the players' backgrounds. We address this problem by creating a game geared specifically to cybersecurity beginners; in fact, to students who may not have any computer science background at all. Our goal is not to educate the next generation of security professionals, but to inform lay people about online risks, spark interest in the topic and motivate learning more.

## 2 Related Work

Perhaps the most well-known cybersecurity game is the *CyberCIEGE* game. Players assume the role of an IT decision maker for a small business in a 3D office environment. Scenarios challenge the players to make security decisions and depict realistic tradeoffs and risk management. The scenarios educate about concepts such as encryption, DMZ, and patches [5]. *CyberCIEGE* has been tested in a number of computer security courses (e.g. [10]).

Another cybersecurity game, *Cash City*, teaches about cybersecurity in a digital Monopoly-style game. The game was evaluated on first-year IT students and showed modest improvement for the game-playing group over a control group [6]. [d0x3d!][1] is a tabletop card game that is open-source and available for downloading and remixing. It has been assessed in a number of field tests and has received positive feedback [2]. Several games have been created to specifically focus on phishing awareness, e.g. [9, 11, 1]. A survey of cybersecurity games, both those reported about in academic literature and those created by private industry, is discussed in [4].

[1] `http://d0x3d.com/d0x3d/welcome.html`

## 3 The Game

In the game, which is loosely based on the Game *Spent*[2], the player has been hired as an IT specialist. S/he must then navigate through routine challenges and learn along the way: each "month" in the game contains specific learning modules which the player must successfully navigate, based on the learning goals summarized in Table 1. Players are given a brief tutorial and then have to use the information to succeed at a related quiz or challenge: for example, crafting a password and then seeing how strong it is, determining whether an email is a phishing scheme, encrypting and decrypting using a variety of encryption methods, and making various security choices for the company (e.g. choice of backups for sensitive data), as well as short multiple-choice quizzes.

Table 1: Learning Goals of the Game

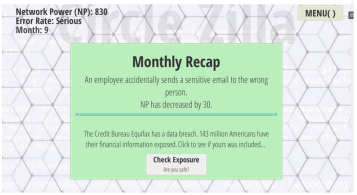
| | |
|---|---|
| passwords | creation of passwords that are robust against dictionary attacks |
| data backups | importance of, different methods and advantages and disadvantages |
| phishing | ability to discern between safe and unsafe emails |
| malware | basic types of malware, characteristics, and what to do |
| encryption | basic idea, Caesar cipher & drawbacks, one-time pad and RSA |

Along the way, the player is also informed about random events that happen to the company, both negative and positive. In addition to making the game more fun, the events serve as extra learning tools. For example, the hard drive may fail, and the impact of that event will depend on whether the player had previously selected to backup the data. Similarly, players who have chosen Dropbox as cloud storage may be randomly informed of a Dropbox hacking that steals their data. Some of the random events include: firmware updates to patch security holes; hard drive failure; a simulated Equifax data breach.

Player success is calculated through a combination of "network power," which is the quantitative scorekeeping system, and error rate. The error rate influences the probability of negative events happening to the player and can be decreased through successful completion of learning challenges. To be extra-friendly to novices, we allow them to select the challenge level of the game (Figure 2a). An "IT Senior" provides advice as necessary (e.g. Figures 2c and 2d). The game is published as a WebGL, so that it is playable in the browser with no need for installation.

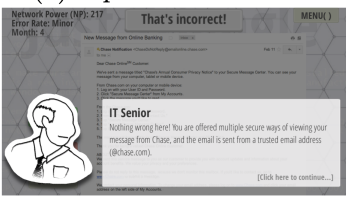
[2] `http://playspent.org/`


(a) Choose challenge level




(b) Equifax data breach


(c) IT Senior provides hints




(d) IT Senior corrects mistakes

Table 2: Game pictures

# 4 Impact of the Game

## 4.1 Overview

*Cyber Secured* was piloted in the Electronic Commerce course offered at our US urban public college. The course is co-listed under the Computer Science and Business departments and the students are fairly evenly divided between the two departments. Most of the CS students (and typically all of the Business students) are new to the field of computer science, and specifically to cybersecurity.

The course covers the basic technological and business background of e-commerce, including the development of the Internet and the WWW, business strategies, cybersecurity, and marketing. In the cybersecurity portion of the course, the goal is to give a general overview of the insecurity of the Internet, educate about security threats and explain the high-level concepts behind encryption. *Cyber Secured* was designed to help teach these basic topics to this pool of beginners.

To determine the potential of *Cyber Secured* to improve students' knowledge of cybersecurity and to get student feedback about the game, students taking the e-commerce course were offered the opportunity to play *Cyber Secured* for extra credit on a homework assignment. In total, 118 students were presented with this offer, comprising three sections of the e-commerce course, one online and two traditional in-class courses: an in-class section in the Spring (hence-

forth, Class1), and an in-class section (henceforth, Class2) and an online section (henceforth, Online), both in the following Fall. The students were briefed on the study and informed consent was obtained. They were then asked to take a pre-test measuring their knowledge of basic cybersecurity concepts, play the game, and take a post-test. The pre- and post-tests consisted of 13 questions covering passwords, phishing, malware, and encryption; to save space, we give only a sample of the questions in Figure 1. Questions on pre- and post-tests were highly similar. The pre- and post-tests were administered through Google Forms. The game was posted online[3].

Figure 1: Sample post-test questions (correct answers are underlined)

1. Which of these passwords is the strongest?
   (a) ILoveSchool!
   (b) hello!8
   (c) monkey
   (d) YaThink?
   (e) I don't know

   Briefly explain your answer.
2. A phishing email is:
   (a) an email that has a virus inside
   (b) an email that tries to solicit sensitive information from you
   (c) an email that has spyware attached
   (d) an email that automatically gets sent to all of your email contacts
   (e) I don't know
3. If you're unsure if an email from a specific site is a phishing attempt, you should
   (a) click on the link provided
   (b) type the URL of the site directly into your browser
   (c) reply to the email to see if it bounces back
   (d) open the attachments provided
   (e) I don't know
4. Encrypt the following text, using a Caesar cipher with a key value of 2: `hello`. jgnnq
5. Decrypt the following text, using a Caesar cipher with a key value of 3: `fbehu`. cyber.
6. Is a Caesar cipher a strong encryption method? Explain your answer briefly.

Some changes were made to the game between the Spring and Fall semesters, including adding content about one-time pads and RSA encryption. To keep this study consistent between the cohorts, we used the same pre- and post-tests in the Fall semester as in the Spring, omitting questions about the new content.

[3] `https://cybersecured.itch.io/cyber-secured-2020`

### 4.2 Participants

In Class1, 40 students were offered the opportunity to participate. Of the 40 students, 23 took the pre-test, played the game and took the post-test (12 male and 11 female). We denote this group as Game (G). The other 17 students are our Control group (C); of these, 14 did not take the pre-test or play the game and three took the pre-test but did not play the game or take the post-test. Class 2 had 40 students. Of these, thirty chose to participate by playing and taking both tests (Game, 21 male and 9 female) and ten students did not take the pre-test or play the game (Control). Finally, of the 38 students in the Online course, 22 were in group Game (12 male, 10 female) and 16 did not take the pre-test or play the game (Control).

### 4.3 Pre-test to Post-test

We give the mean and median for both tests for the 75 students who played the game below in Table 3. Both the pre- and post- tests were scored out of 13 points, and the scores are given both as raw scores and as percentages. Scores are presented for each of the individual sections as well as the combined group of all students. The average scores on the post-test demonstrate statistically significant increases compared to the pre-tests (using a paired t-test, $\alpha = .05$, $p < .001$ for all three sections).[4]

Table 3: Game Quizzes, Group Game

| | | Average Score | Median |
|---|---|---|---|
| Class1 ($n = 23$) | Pre-test | 7.7/13 (59%) | 7.3/13 (57%) |
| | Post-test | 10.6/13 (82%) | 12/13 (92%) |
| Class2 ($n = 30$) | Pre-test | 9.0/13 (69%) | 9/13 (69%) |
| | Post-test | 10.6/13 (81%) | 11/13 (85%) |
| Online ($n = 22$) | Pre-test | 6.0/13 (46%) | 6/13 (46%) |
| | Post-test | 8.2/13 (63%) | 8/13 (62%) |
| Combined ($n = 75$) | Pre-test | 7.7/13 (59%) | 8/13 (62%) |
| | Post-test | 9.9/13 (76%) | 10/13 (77%) |

### 4.4 Final Exam Scores

To measure retention and transference, we looked at performance on the final exam. All sections of the course had questions on the final exam that related to the cybersecurity concepts covered in the game (e.g. malware, phishing, and encryption). The two Class groups had similar finals, with the questions

[4]We note that the scores on the pre- and post-tests were significantly less for the Online group than the corresponding scores in both Class groups ($p < .01$ for all). We do not offer conjectures to explain this, particularly due to such a small sample size.

about security (Class Security-Questions, or C-SQ) worth 26 points total; the online final had Online-Security Questions (O-SQ) worth a total of 17 points. The questions on the final give us a way to measure the impact of the game on longer-term retention and knowledge, despite not corresponding closely to the questions in the pre-/post-tests.

In Table 4, we give the average scores of the pre-tests and the security questions on the final for each section, given as percentages of total possible points. We also calculated the difference between the pre-test score and the score on the final security questions; the average of those differences is given in the last column on the table. We analyzed the pre-test score and final SQ score for each student in the Game groups using a paired t-test. The final SQ scores were significantly greater than the pre-test scores ($p \leq .001$) for each Game group, indicating retention of the material taught (though, of course, other factors may also have contributed).

Table 4: Average Pre-test and SQ Scores

| | Pre-test | Final SQ | Difference |
|---|---|---|---|
| Class1 ($n = 23$) | 59% | 76% | +16% |
| Class2 ($n = 30$) | 69% | 79% | +10% |
| Online ($n = 22$) | 45% | 77% | +32% |

We also compared the final SQ scores of those who played the game to their classmates who did not. In Table 5, we give the average scores for the security questions (SQ) for the Class and Online groups. The last column gives the grade for the "Rest of the Final" (ROF): the non-cybersecurity questions on the final (as a percentage of the remaining 74 and 83 points, respectively), to serve as a control group for the SQ.

In both the Class and the Online Control groups, the scores on the SQ lag considerably behind the scores on the ROF. This is consistent with our observation that students find cybersecurity of the most difficult topics in the course. In contrast, scores on SQ for the Game groups were similar to their ROF scores, in addition to being significantly higher than Control-SQ scores. This suggests that the game helped students understand and retain the material.

However, it is difficult to draw any concrete conclusions about the effect of the game. The differences in scores for the SQ were not statistically different between the Game groups and Control groups for the Class groups and likewise, the ROF scores were also not significantly different. In the Online group, on the other hand, there was a statistically significant difference between the SQ scores of Game and Control groups, but also a statistically significant difference between ROF scores ($p < .001$, $p = .003$, respectively). Hence, while we find the results of the final questions encouraging, we see possible evidence of a selection bias indicating that the differences in SQ scores may have been (at least partially) caused by underlying differences in the Game and Control

groups. It is also possible that our small sample size does not allow us to adequately study the effects on the final.

Table 5: Average Scores on Final SQ

| | SQ | ROF |
|---|---|---|
| Class1: Control($n = 17$) | 63% | 73% |
| Class2: Control ($n = 10$) | 63% | 67% |
| Combined Class Control: ($n = 27$) | 63% | 70% |
| Class1: Game ($n = 23$) | 76% | 79% |
| Class2: Game ($n = 30$) | 79% | 77 % |
| Combined Class Game: ($n = 53$) | 77% | 78% |
| Online: Control($n = 16$) | 53% | 66% |
| Online: Game ($n = 22$) | 76% | 77% |

### 4.5 Interest in Cybersecurity

In addition to increasing *knowledge* about cybersecurity, another goal of our game was to increase *interest* in cybersecurity, with the goal of students finding the topic intriguing and relevant. We attempted to ascertain whether that goal was met. All three finals contained a two-point question that asked "*In your opinion, what was the most interesting / informative / useful topic that we covered? Briefly explain your answer.*" This question solicits general feedback about topics that interested students. We use the responses to estimate student interest in cybersecurity.

Table 6: Cybersecurity response rate

| | Control | Game |
|---|---|---|
| Class1 | 44% | 54% |
| Class2 | 25% | 56% |
| Online | 23% | 53% |
| Combined | 33% | 54% |

Out of the 118 students in the three sections of this course, 108 responded to the question. For each of the responses, we tallied up which students chose cybersecurity, or any of its sub-topics (e.g. malware, phishing, encryption) vs. other topics in the course. We give the cybersecurity response rates for each section in Table 6. In each section of the course, more than half of the Game group chose cybersecurity as their "favorite" course topic. The same was not true of the Control group, whose cybersecurity responses were much less frequent. The difference in the response rate was statistically significant ($p = .04$). We see this as an indication that playing the game increased interest in cybersecurity. (Some of the Game responses made that explicit, e.g. "*The game really helped me delve into the topic.*")

We also looked at response rates to this question from a previous semester, in which cybersecurity was taught but the game was not offered as a resource.

That rate gives a baseline of cybersecurity interest among our students. The response rate of cybersecurity topics in the past was 38%; the difference between that rate and the response rate of the Game groups was statistically significant ($p = .049$). This suggests that the game actually increased interest in cybersecurity, and that the effects that we observed between Game and Control groups were not merely selection bias.

### 4.6 Qualitative Survey Results

The post-test also included qualitatives survey questions. The first four were based on [3] to measure the levels of intrinsic motivation of the students. The second three questions measured the students' engagement with the game. All seven questions used a 5-point Likert scale, labeled from "strongly disagree" (1) to "strongly agree" (5). An additional two questions asked for students' feedback on the game. The results of the rating questions are shown in Table 7. (To save space, we condense the intrinsic motivation questions.) In the second column, we give the average scores, across all sections. In the third column, we give the percentage of responses that indicate agreement; i.e. either "agree" or "strongly agree" ($\geq 4$).

Table 7: Qualitative Survey Responses

| Question | average | percent agreeing |
|---|---|---|
| I played this game because I found it interesting | 3.6 | 55% |
| I would play this game for fun | 2.9 | 29% |
| I would play this game to learn about security | 4.3 | 82% |
| I would play this game to help assess my knowledge of security | 4.2 | 82% |

The average score across all four intrinsic motivation questions was 3.3, indicative of slightly above-average motivation. Although students were neutral about the "fun" qualities of the game, the responses indicate strong agreement with the educational and assessment qualities of the game; over 80% of the students said that they would play this game to learn about and assess their knowledge of security.

The survey also included two-open ended questions for feedback: *Please tell us at least two things that you did not like about the game or think should be changed* and *Please tell us at least two things that you liked about the game – things we should not change.* These questions were not required fields; 67 (89%) of students chose to answer them. We hand-tagged the comments to identify common themes between answers; comments could be tagged with multiple tags. In total, we identified 16 "don't like" and 15 "like" tags. On the "like" end, 45% of students who answered commented about the educational benefits of the game, with comments such as *"I like that it taught you about the different topics while having fun; A great way to learn about the topic. Different in a*

*good way; The content was very informative!; The game was very interesting and informative. It would be great if there were more games like this to teach students about the topics in e-commerce."* 18% of the responses commented on the fun aspect of the game: *"I enjoyed playing until the 12 months were over; It felt very much like a regular game; The game was creative and unique unlike most browser games."*

On the "don't like" side, 15% of the responses noted that the game was confusing in some way (*"Make the instructions a bit clearer; the game error level was confusing at first"*); 12% wanted the graphics to be improved (*"I wish the game was a bit more colorful; the graphics could be improved"*) while 12% could not think of any improvements necessary (*"Nothing that I didn't like; None. I liked the game"*). The most frequent comment (16%) was that the game was too long (*"The game felt long; it was kinda long; I think the game is a little bit too long to play"*). As noted in Section 4.1, we added two new encryption modules between Class1's participation and that of Class2/Online. Responses about length were far more prevalent in Class2/Online surveys, and slightly lower levels of interest in the game were reported in those surveys as well. We suspect that the additional modules may have made the game too long for students and decreased their interest in playing. One of our plans is to make these modules optional, so that students can choose to skip them.

## 5 Discussion and Conclusions

Although our sample size is small, our results are encouraging and show that students who play *Cyber Secured* demonstrate educational gains compared to students who did not play. Moreover, student feedback suggests that students themselves recognize the value of the game as a tool for learning about cybersecurity and several students asked for us to create similar games for other course topics. We think that this game has potential to be helpful to students who are new to cybersecurity and plan to continue to develop the game by making some modules optional as well as by making the game more visually appealing and fun to play by speeding up slow-loading text, clarifying instructions and improving the graphics. We would also like to add analytics so that we can see with which topics students struggle, and we plan to then conduct a larger study of its effects on a larger sample of students.

This game was created by a team of undergraduate CS students. This project was doubly enriching, offering educational benefits for both the students who created the game and those who played it. Besides improving their skills in Unity game programming, the students who developed the game learned to manage a complex code base, to work as a team, and to design a pleasing user experience. This game is thus a strong representation of "of students for students."

The game is available for free online at `https://cybersecured.itch.io/cyber-secured-2020` as a resource for other instructors who teach cybersecurity to novices. Because of the introductory nature of the game, it can be used in a variety of courses, including General Education courses, CS0 courses and other CS courses for non-majors. The game can also be a meaningful addition to high school CS courses. *Cyber Secured* can be used as a standalone course activity to raise awareness about cybersecurity, as a means of assessing student knowledge, or as an introduction to the topic. It can also be used to motivate class discussions about cybersecurity, as well as debates about the ethical issues that surround cybersecurity.

## References


[1] Nalin Asanka Gamagedara Arachchilage, Steve Love, and Konstantin Beznosov. Phishing threat avoidance behaviour: An empirical investigation. *Computers in Human Behavior*, 60:185–197, 2016.

[2] Mark Gondree and Zachary NJ Peterson. Valuing security by getting [d0x3d!]: Experiences with a network security board game. In *6th Workshop on Cyber Security Experimentation and Test*, 2013.

[3] Frédéric Guay, Robert J Vallerand, and Céline Blanchard. On the assessment of situational intrinsic and extrinsic motivation: The situational motivation scale (SIMS). *Motivation and Emotion*, 24(3):175–213, 2000.

[4] Maurice Hendrix, Ali Al-Sherbaz, and Bloom Victoria. Game based cyber security training: are serious games suitable for cyber security training? *International Journal of Serious Games*, 3(1):53–61, 2016.

[5] Cynthia E Irvine, Michael F Thompson, and Ken Allen. CyberCIEGE: gaming for information assurance. *IEEE Security & Privacy*, 3(3):61–64, 2005.

[6] Thomas Monk, Johan Van Niekerk, and Rossouw von Solms. Sweetening the medicine: educating users about information security by means of game play. In *Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*, pages 193–200. ACM, 2010.

[7] Marina Papastergiou. Digital game-based learning in high school computer science education: Impact on educational effectiveness and student motivation. *Computers & Education*, 52(1):1–12, 2009.

[8] Portia Pusey, David H Tobey, and Ralph Soule. An argument for game balance: Improving student engagement by matching difficulty level with learner readiness. In *3GSE*, 2014.

[9] Steve Sheng, Bryant Magnien, Ponnurangam Kumaraguru, Alessandro Acquisti, Lorrie Faith Cranor, Jason Hong, and Elizabeth Nunge. Anti-phishing Phil: the design and evaluation of a game that teaches people not to fall for phish. In *Proceedings of the 3rd symposium on Usable privacy and security*, pages 88–99. ACM, 2007.

[10] Michael Thompson and Cynthia Irvine. Active learning with the CyberCIEGE video game. In *Proceedings of the 4th Conference on Cyber Security Experimentation and Test*, 2011.

[11] Shian-Shyong Tseng, Kai-Yuan Chen, Tsung-Ju Lee, and Jui-Feng Weng. Automatic content generation for anti-phishing education game. In *Electrical and Control Engineering (ICECE), 2011 International Conference on*, pages 6390–6394, 2011.

[12] Pieter Wouters, Christof Van Nimwegen, Herre Van Oostendorp, and Erik D Van Der Spek. A meta-analysis of the cognitive and motivational effects of serious games. *Journal of Educational Psychology*, 105(2):249–265, 2013.

# Introducing Programming using Previewing*

*Chris Alvin*
*Computer Science Department*
*Furman University*
*Greenville, SC 29613*
*calvin@furman.edu*

**Abstract**

We want students to begin taking ownership of programming language concepts in a CS1 class as soon as possible; however, it can be difficult to provide a meaningful and interactive activity on the first day of class. We describe a programming activity that attempts to immerse students in source code as a way to introduce basic programming concepts as well as bridge their prior knowledge with future course content. This paper introduces the idea of a *pre-programming* activity to the computing community, describes our interactive laboratory activity, and describes short-term analyses of the effectiveness of our activity.

## 1 Introduction

The process of learning to program may be described by two core components: (1) developing an algorithmic solution and (2) expressing an algorithm in a language foreign to the student. Individually and collectively, these tasks are challenging in a CS1 course setting. For a student with no programming experience, a first course in programming may be intimidating. While it is our goal as computing educators to engage students in algorithmic problem solving, it is also our goal to demystify the idea of source code. Simply, we want students to write transparent source code for themselves and for others. As part of this process, we also want students to understand that documenting source

code through comments is a strong communication tool that can illuminate an algorithm.

In this work, we discuss a first laboratory activity for CS1 that serves as an immersive, "previewing"-style learning device that can be administered on the first day of class. *Previewing* (including, 'pre-reading') is a well-known concept in K-12 education and can be described as *any activity performed before reading that is meant to assist a learner in understanding a text.* Such preparation activities are what help a learner connect new information with what they already know; hence, learners are more likely to take ownership of those new concepts. Pedagogically, our *pre-programming* activity has several goals:

- Introduce basic functionality of the Integrated Development Environment (IDE);
- Introduce select programming vocabulary and link that vocabulary to existing knowledge;
- Engage students in syntax and semantics of the target language while reinforcing mathematical norms of the language;
- Introduce control flow with selection and functions for modularity;
- Illustrate the level of detail required in source code;
- Demonstrate the importance of communicating with the reader (and yourself) through rich, yet reasonable comments; and
- Convey that source code may be understood using a combination of good documentation and precise code statements.

These goals are lofty; however, we show that they are attainable for most students using a pre-programming activity.

## 2 The Pre-Programming Activity

**Motivation.** In our experiences, a typical introductory programming class consists of students with a wide variety of backgrounds. Even students who have taken Advanced Placement courses in high school evidence a wide variance in their knowledge of syntax and semantics of Java or another programming language. In a CS1 course, most students are not often able to communicate well (written or verbal) efficient and/or effective algorithms. This variability in student background can help to create an interesting dynamic in the classroom, both positive and negative. For example, it is easy to understand when a student with some background in programming excels early in an introductory programming course, but loses steam with more complex material later in the course.

We use Python in our CS1 course. Since Python deviates sharply from the

syntax and semantics of Java and other programming languages students may have experienced prior to the course (e.g., Javascript, R, C#, etc.), experienced students often have much to learn on the first day of the course. While introducing aspects of programming to students is important, our pre-programming activity immediately engages all students using a common activity.

We have three overarching goals for our pre-programming activity. First, it provides a common experience for all students in the course; we can thus make continued reference to it for many weeks into the course. Our second goal is to provide a balanced knowledge bridge into course content for students with or without prior programming knowledge. Last, but not least, the activity introduces some fundamental programming concepts as well as a sample of programming language constructs.

**The Activity.** The pre-programming activity [1] consists of 49 questions. Many of the questions are framed as (1) carry out an operation and (2) report the result. Several other questions ask students to generalize their findings. The activity takes between 45-120 minutes depending on prior student experience, comfort with new software, (mis)interpretation of instructions, etc. When administered, an instructor and a lab aide were available for guidance.

*IDE.* The activity begins with an overview of the IDE. Using an annotated screenshot of the IDE interface, we describe the editor and shell (or console). As a segue, the activity mentions that Python is an interpreted language (without further description) thus facilitating interactivity with the shell: "Hence, you can enter an instruction into the shell and it will execute immediately."

*Evaluating expressions.* To engage students as quickly as possible, the next sequence of questions asks them to investigate operators. Students are asked to evaluate `2**4` in the shell and then asked to infer the function of the `**` operator. Students then interact with `+` applied to strings by evaluating `"Sm"+"ile!"` and `'Fro'+'wn.'`. Students are asked if there seems to be a distinction between single quotes (') and double quotes ("); we find this detailed question important to distinguish Python from Java for students with experience. We then combine the idea of mathematical and string expressions by having students evaluate `print` statements such as `print(3*4-5)` compared to `print("3*4-5")`. Our goal in introducing these concepts is not complete coverage, but to provide an intuitive introduction to these fundamentals concepts.

*Indentation.* To demonstrate the importance of indentation in Python, we present the code in Figure 1 [1], but with all vertical whitespace and all leading indentations removed. It is important to introduce students to Python error messages in our IDE: "expected an indented block" (Line 9 in the unindented code). However, students also begin to see, by example, that indentation in the Python language is critical to program execution. Last, students may recognize the difficulty in interpreting dense code in a holistic manner. That is,

```
import math
def roots(fltA, fltB, fltC):
    """Computes and prints the roots of a quadratic polynomial: ax^2 + bx + c
        @input: fltA -- a in the expression ax^2 + bx + c
        @input: fltB -- b in the expression ax^2 + bx + c
        @input: fltC -- c in the expression ax^2 + bx + c
        @output: No output; the result is printed
        @Restrictions: None"""

    # Compute the discriminant of the quadratic formula
    fltDiscriminant = fltB * fltB - 4 * fltA * fltC

    # Check if roots are imaginary: no roots
    if fltDiscriminant < 0:
        print ('None')
        return

    # One unique root
    if fltDiscriminant == 0:
        print ('x =', -fltB / (2 * fltA))
        return

    # Compute the two roots
    fltX_1 = (-fltB + math.sqrt(fltDiscriminant)) / (2 * fltA)
    fltX_2 = (-fltB - math.sqrt(fltDiscriminant)) / (2 * fltA)

    # Report two roots
    print ('x1 =', fltX_1)
    print ('x2 =', fltX_2)

def test():
    # Call the function with three input values: a, b, c
    floatA = 1.0
    floatB = -2.0
    floatC = -3.0
    print(str(floatA) + "x^2 + " + str(floatB) + "x + " + str(floatC) + ":")
    roots(floatA, floatB, floatC)

# Invokes the testing function
if __name__ == "__main__":
    test()
```

Figure 1: Version 1 of the `roots` function source code.

reading code without 'quality of life' indentation and whitespace can make code needlessly complex to parse, understand, and modify.

*Code comprehension.* The rest of the lab is based on two versions of code that computes and prints roots of a quadratic polynomial via the quadratic formula (Figure 1 and Figure 2 [1]); for brevity, we omit some calls to `roots` in `test` in Figure 1 and all invocations of `roots` in Figure 2. We chose an algebraic concept since, in theory, it would be recognizable to students although we understand some student hesitancy toward mathematics. In the activity, we remind students that the quadratic formula ($x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$) is used to solve quadratic polynomial equations of the form $ax^2 + bx + c = 0$ where $a, b, c$ are real numbers as well as the fact that the *discriminant* refers to the expression

under the radical ($b^2 - 4ac$).

Students import and execute the code in Figure 1 and then are asked to consider the following concepts by making temporary or permanent changes to the code. We list some of the concepts here; later, we describe a few in greater detail.

- Comments with `#` and docstring comments with `"""`.
- Legal string literals by balancing `"`.
- Floating-point numbers compared to integers.
- The absolute value function `abs`.
- Using vertical and horizontal whitespace for clarity: (e.g., `2 * a` versus `2*a`).
- Modules, in particular, the `math` module.
- Escaped characters, including tab and newline.
- Line by line code traversal with different input values.
- Function parameters as local variables.
- Control flow with selection statements.

*Comments and* `help`. In a sequence of questions, we ask students to intuit the purpose of comments as well as commenting syntax in Python. In particular, we ask students to remove a `#` character, report any problems, and speculate as to whether the interpreter ignores the succeeding text. Students then make a conjecture as to the purpose of comments.

Students then explore docstring comments; our style is influenced by Java docstring comments. We have found that enforcing this format communicates the critical elements of a function and provides a clear segue into the next course (which teaches Java). Students explore one use of docstring comments by using the built-in Python `help` function in the shell (`help(roots)`). Further, students consider the predefined absolute value function (`help(abs)`). Students were quick to comment that "abs(x, /) Return the absolute value of the argument" was not necessarily clear to new programmers because of the term *argument*.

*Modules.* A quadratic formula implementation requires the square root function. To introduce the idea of library functionality and implementation hiding, we ask students to modify the code in Figure 1 to remove `import math`. As described in the activity "With one small change, our Python code is now attempting to use a function (`sqrt`) for which it cannot find its definition." Students then update the `sqrt` function call to `math.sqrt`; this exemplifies that a programmer needs to specify where functions are defined.

*Exploring control flow.* The remaining 12 questions engage students with the code in both Figure 1 as well as a fully-commented version of the code in Figure 2. The idea of control flow and selection is introduced by having students modify input values to `roots` and assessing the output. Students consider input

```
1  import math
2  def roots(fltA, fltB, fltC):
3      fltDiscriminant = fltB * fltB - 4 * fltA * fltC
4
5      if fltDiscriminant < 0:
6          print('None')
7      elif fltDiscriminant == 0:
8          print('x =', -fltB / (2 * fltA))
9      else:
10         fltX_1 = (-fltB + math.sqrt(fltDiscriminant)) / (2 * fltA)
11         fltX_2 = (-fltB - math.sqrt(fltDiscriminant)) / (2 * fltA)
12         print('x1 =', fltX_1)
13         print('x2 =', fltX_2)
```

Figure 2: Version two of the `roots` function (listed without comments).

values $(1, 0, 0)$ (corresponding to $x^2 = 0$) resulting in the output $x = 0$. In a second case, they are asked to construct input values corresponding to $x^2 + 1$ (with output *None*). Last, by mathematical background (or trial and error), students were tasked with defining an input tuple of values in which `roots` outputs two unique roots. From this case-based analysis of distinct input values, students were asked to consider the semantics of the `if` and `return` statements by analyzing the resulting selective control flow.

We then asked students to consider a well-commented version of the code in Figure 2 that implements a version of `roots` without explicit `return` statements. Experienced programmers may observe that both implementations are case-based selection logic and thus are quite similar (although their corresponding control-flow graphs are distinct). Hence, the preferred implementation comes down to programmer style, comfort, and choice. Beyond basic selection, we have students consider second version of `roots` to introduce the `elif` and `else` keywords, show that there are multiple, acceptable ways to implement a solution, and that programmers have contrasting opinions about which version they prefer to read, implement, debug, or update.

## 3 Brief Analysis of the Pre-Programming Activity

**Background: Course and Students.** Our CS1 course is traditional in that it emphasizes algorithms and structured programming. The course seeks to introduce standard control structures (e.g., selection and repetition) and Python data structures (lists and dictionaries). We introduce modular software development practices via functions and modules early in the course with continued emphasis throughout the course. The class consists of a dedicated 2-hour lab period and two 75-minute class meetings per week. Labs are moderated by the instructor as well as a lab aide facilitating individual attention and interactions.

We administered the pre-programming activity [1] to two sections of our

Table 1: Student information from Fall 2019 introductory programming courses.

| Level | | Major | | Background | | Background Language | |
|---|---|---|---|---|---|---|---|
| First Year | 1 | CS/IT | 4 | AP Exam | 2 | Python | 8 |
| Second Year | 8 | Dual with CS/IT | 8 | Non-CS Prereq | 14 | Javascript, NetLogo | 9 |
| Third Year | 6 | Dual without CS/IT | 2 | CS Prereq | 16 | None | 15 |
| Fourth Year | 16 | Not CS/IT | 12 | | | | |
| Non-Degree Seeking | 1 | Undecided | 5 | | | | |
| | | No Major | 1 | | | | |
| Total | 32 | | 32 | | 32 | | 32 |

course in Fall 2019; both sections were taught by the same instructor. In total, the two sections had 32 students; see Table 1 for some background student information including: majors, levels, and background. There are three possible pre-requisite paths into the course. Traditionally, most students take a thematic, general education, survey course for majors and non-majors that is offered each semester with a theme chosen by the instructor. For Fall 2019, 50% of the 32 students took this pre-requisite course. A student may also enter the course by scoring a 4 or 5 on the AP Computer Science Principles exam (students receiving a 4 or higher on the AP A exam receive credit for the course). The third path into the course is for a student to complete a Calculus course or a science course for majors in other areas (e.g., Biology, Chemistry, etc.).

Based on course structure, timing, and allowable pre-requisites, there are few first year students in the course as shown in Table 1. This can be explained by the fact that (1) this course was taught in the fall semester, (2) first year students are encouraged to take a liberal arts schedule that includes other general education requirements, and (3) our program is organized such that the traditional CS1 course is not the first course in the Computer Science department. Although it is not stated in the table, we note 8 of 32 students are either mathematics or applied mathematics majors: 7 of the 8 entered the class without the thematic pre-requisite course.

**Questionnaire.** We administered the same set of questions shown in Figure 3 immediately before and immediately after students completed the pre-programming activity. We are using the questionnaire, in part, to assess the effectiveness of the activity; however, this is not our foremost goal. In previewing terms, administering the questionnaire prior to the activity seeks to activate and build background knowledge in order to enhance comprehension. That is, while the questions may seem foreign to many students before engaging in the activity, it begins the cognitive recognition and recall process. Even

**1.** T / F. Python is an interpreted language.

**2.** T / F. In Python, we must pay attention to indentation of code.

**3.** T / F. State your opinion: it is possible to read and understand the purpose of a block of source code without knowing how to program.

**4.** Evaluate `2 ** 4`.

**5.** State the output of `print(7 - 4 + 3)`.

**6.** State the output of `print("7 - 4 + 3")`.

**7.** How do we indicate comments in Python?

**8.** T / F. Adding vertical whitespace will alter the output of code.

**9.** Evaluate `sqrt(4)`.

**10.** What character does `\t` represent?

**11.** Write a line of code to assign the variable `intA` to be the value `5`.

**12.** T / F. Every line of code in a program will be executed when it is run once.

**13.** Describe the function of an if-statement.

Figure 3: Pre-programming questionnaire administered before and after the activity.

Table 2: Pre- and post-questionnaire student responses for select Figure 3 questions.

| | **1** | **2** | **4** | **5** | **6** | **7** | **8** | **11** | **12** | **Total** | **≈ %** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Corr → Corr** | 19 | 26 | 9 | 21 | 12 | 4 | 17 | 4 | 17 | **129** | **44.8** |
| **Corr → Inc** | 0 | 0 | 1 | 2 | 1 | 1 | 1 | 6 | 4 | **16** | **5.5** |
| **Inc → Corr** | 11 | 6 | 22 | 6 | 10 | 22 | 12 | 6 | 7 | **102** | **35.4** |
| **Inc → Inc** | 2 | 0 | 0 | 3 | 9 | 5 | 2 | 16 | 4 | **41** | **14.2** |

if a student is unsure of an answer, the student is forced to strongly consider the semantics of the question as well as the semantics of the programming language construct.

As shown in Figure 3, our questionnaire consists of 13 questions that attempt to highlight some of the salient ideas that students will encounter in the activity as well as in the course. We attempted to write a set of questions that highlight fundamental programming concepts covered in the activity that are expeditious to complete. We did not want students to be exhausted from the idea of data collection on top of completing the activity. Hence, most of the questions have succinct answers except the last question requiring a sentence or two as a response.

**Analysis.** In our administration of the questionnaire and activity we identified a set of non-ambiguous questions; the results of these questions are summarized in Table 2. We observe that approximately 45% of students maintained correct answers from the pre- and post-questionnaire either from prior knowledge or quality guessing. We also see 35.4% of responses going from incorrect answers to correct answers for a total of $231288 = 80.2\%$ correct responses in

the post-questionnaires.

In contrast, we consider some of the incorrect responses (19.8% overall). What is somewhat concerning is the 5.5% of responses that were correct prior to the activity and incorrect after the activity. We observe that the main questions of confusion may be considered the more difficult questions for students. To gain insight into this issue, we looked more closely at results from questions 11 and 12.

For question 11 about writing an assignment statement, of the 6 incorrect students, 4 had no indicated prior programming experience and 2 had non-Python experience. We observe that 5/6 of those students are majoring or intend to major in a computer science or a math-related field and thus may intuit a correct answer (`intA = 5`) on the pre-questionnaire. Some incorrect responses include `def intA(5)` where a student confused assignment with defining a function. Another notable response confused assignment with console output: `print("intA is", 5, ".")`. All of these responses to question 11 indicated students were making the question overly complex by answering with new tools and concepts from the activity. We also intend to change the wording of the question to be more crisp: "Write a line of code that assigns the value `5` to the variable `intA`".

For question 12, 2 out of the 4 had no programming experience and 2 out of the 4 had other programming experience; these data do not allow us to speculate as to an explanation for incorrect responses.

Questions $3, 9, 10$, and 13 from the questionnaire in Figure 3 were not included in Table 2 because we considered them to be either opinion-based, ambiguous, or answers were ambiguous.

We posed question 3 as a means of indicating to students that, while source code may seem foreign, it can also be written in a meaningful way. We found that 10/32 said it was not possible to understand a block of code without programming knowledge. After the activity, 6 of those 10 reported that their opinion had changed indicating it is possible to understand source code. Contrasting, 3 of the 22 reversed their opinions in that it was not possible for the uninitiated to interpret source code. Overall, we are pleased that students believe comprehension of code is possible even with a limited background.

Question 9 (evaluating `sqrt(4)`) seems straightforward: 28/32 students had a 'correct' answer of 2.0 (or 2) in the pre-questionnaire. However, after the activity, 11 answered with the value 2 while 18 suggested the instruction resulted in an error since it omitted the `math` module. We deemed the question to be ambiguous considering how the activity introduces and discusses modules; we intend to edit the question to include a phrase about how `import math` is assumed (or not). Regardless, we are pleased with the student retention of restrictions that modules have on code.

On its face, question 10 seems to be a decisive question about the tab character. In fact, 21 post-questionnaire responses indicate correctly the character is a tab. However, another 10 indicated it represented “spaces” or “indentation”. While these responses are incorrect, they do indicate an intuitive understanding of the concept.

Question 13 is an open-ended question intended to elicit student feedback by explaining a programming concept in their own words. Responses were varied and are relatively difficult to compare pre- and post-questionnaire; we will focus on post-questionnaire responses. For example, “Runs lines of code only under certain conditions." is a response that shows clear evidence of student ownership of the concept. Compared to the response "When you have a condition, if there are multiple options, you can only execute 1." is a student showing understanding, but is referring to a particular implementation: `roots` in Figure 2. Other post-questionnaire responses indicate a lack of understanding selective execution: "Evaluate something within a criteria." or "To differentiate between lines of code." Our goal in analyzing the responses to this question are not for correct or perfect responses, but instead, to abstract the idea of selection and communicate it in their own words as a means of self-reflection.

## 4 Brief Discussion

We have found our pre-programming activity to be effective for several reasons. It solves the problem about what to do during the first laboratory when students have little programming knowledge. In the two years we have used this activity we have observed anecdotally that the activity works to make students less apprehensive about the course and lessen the psychological barriers students erect related to the difficulty of programming even if the activity has a math-based focus. We feel this activity works to make programming more accessible to students and thus is worthy of continued use and evolution.

The activity also helped students identify *course content schema*: what to anticipate in terms of concrete concepts as well as some of the more abstract algorithmic ideas. A few students summarized this idea in conversation, "If the professor highlighted concepts on the first questionnaire, it is important for me to know."

We understand that the idea of previewing and the use of a pre-programming activity is not new. In fact, we recognize that most faculty engage in such activities in many of their courses, specifically introductory programming courses. However, we found the computing literature to be nearly vacant with the idea of pre-programming. One notable exception is [2] that attempts to shepherd students into programming and effective design using a framework that is language agnostic. We holistically appreciate this text, but note that our pre-

programming activity is targeted toward our IDE as well as the syntax, semantics, and documentation style of our target programming language. We believe that an intentionally targeted previewing activity is a means by which we can reduce the barrier for entry in programming classes as well as increase retention from students of all backgrounds.

## 5 Conclusions

We described a pre-programming activity that seeks to connect student knowledge with forthcoming programming concepts while setting the stage for an introductory programming course. We then analyzed the activity through a brief pre- and post-questionnaire showing more than 80% correctness in post-questionnaire responses. We feel the pre-programming activity needs a new core example as to not intimidate math-phobic students. Informally, we described how this activity serves as a consistent point of reference for CS1 students while also attempting to decrease the psychological barrier to entry for students into programming.

## References

[1] Chris Alvin. Introduction to programming using previewing repository, 2021. https://github.com/wcatykid/IntroProgrammingUsingPreviewing.

[2] Stewart Venit and Elizabeth Drake. *Prelude to Programming.* Addison-Wesley Professional, 6th edition, 2014.

# vWaterLabs: Design and Characteristics of a Virtual Testbed for Water-focused ICS Cybersecurity Education*

*Matthew J. Kirkland[1], Stu Steiner[2], Daniel Conte de Leon[1]*
*[1]Center for Secure and Dependable Systems*
*University of Idaho, Moscow, ID 83844*
*[2]Computer Science Department*
*Eastern Washington University, Spokane, WA 99201*
*kirk8182@alumni.uidaho.edu, ssteiner@ewu.edu, dcontedeleon@ieee.org*

**Abstract**

Industrial Control Systems (ICS) are increasingly being targeted by cyber-attacks while the need for cybersecurity professionals with knowledge of ICS is at an all-time high. Education and training for ICS cybersecurity is costly and not available at the needed scale. In addition, there is a lack of easily available and cost effective material focused on ICS cybersecurity education, which impedes our nation's ability to quickly educate and train ICS cybersecurity professionals. Within this context, water control systems have not been given as much attention from the educational community as other ICS systems such as electric power. To help solve this problem, we created vWaterLabs: a virtual testbed with an accompanying set of labs for ICS cybersecurity education focused on water systems. The contributions of this article are: (1) Analysis of the characteristics of educational and water focused ICS testbeds, (2) Design details of a virtual-only and cost effective testbed for ICS cybersecurity education focused on water systems, and (3) Introduction of an ICS vulnerability assessment lab implemented using the virtual testbed. vWaterLabs is open source and freely available at https://github.com/ICSSecurityLabs.

# 1 Introduction

Malicious threats to Industrial Control Systems (ICS) and critical infrastructures are growing rapidly [3]. In addition to the rise of critical infrastructure threats, there exits a well-documented need for training and education of cybersecurity professionals which may be even greater for the case of ICS cybersecurity professionals.

To meet the demand for cybersecurity professionals, colleges and universities are increasingly offering cybersecurity courses and degrees. Industry is also increasing training opportunities, some targeted to ICS cybersecurity. ICS-CERT, SANS, and a few other organizations offer ICS cybersecurity training, including hands-on experiences taught by leading industry professionals. Despite these efforts, current training and educational opportunities cannot reach the scale needed to meet the demand for cybersecurity professionals within the next few years.

## 1.1 Proposed Solution

We designed and implemented vWaterLabs to help rapidly increase the training and education of cybersecurity professionals with knowledge of water-focused ICS cybersecurity. vWaterLabs is a virtual testbed for industrial control systems cybersecurity education focused on water control systems along with a set of accompanying labs. The objectives of vWaterLabs are:

- Provide a testbed for ICS cyber-education focused on water systems.
- Enable hands-on learning in water ICS cybersecurity.
- Ensure the testbed is cost effective, easily replicated, and easy to use.
- Ensure the testbed and labs help satisfy the Centers for Academic Excellence in Cyber Defense (CAE-CD) relevant Knowledge Units (KUs).

## 1.2 Contribution

The contributions of this article are: (1) Analysis of the characteristics of educational and water focused ICS testbeds; (2) Design details of a virtual and cost effective testbed for ICS cybersecurity education focused on water systems; and (3) Introduction of an ICS vulnerability assessment lab implemented using the vWaterLabs virtual testbed.

A separate article, titled *vWaterLabs: Developing Hands-On Laboratories for Water-focused Industrial Control Systems Cybersecurity Education*, describes the following contributions: (1) Pedagogical analysis of the knowledge and skills needed for water-focused ICS cybersecurity practitioners; (2) Design considerations and decisions for easily reproducible labs for water focused ICS

cybersecurity education; (3) Introduction of a MODBUS injection and mitigation's lab that uses the vWaterLab virtual ICS testbed. This related article is expected to be appear in the Journal of Computing Sciences in Colleges, volume 36, issue 10.

Our ultimate goal is that vWaterLabs will provide instructors and students opportunities for development of clear, detail-oriented, and hands-on understanding of the approaches, techniques, and tools used to protect today's ICS systems related to water processing.

### 1.3 Overview of this Article

The rest of this article is organized as follows: Section 2 describes educational ICS testbed characteristics. Section 3 describes the architecture of a water-focused virtual testbed. Section 4 introduces an ICS vulnerability assessment lab implemented on the virtual testbed. Section 5 presents our conclusion and proposed future work.

## 2 Educational ICS Testbed Characteristics

Using Google Scholar, we searched for and reviewed in detail ICS cybersecurity education literature. For this search, performed in April 2019 and then updated in June 2020, we used the following search string: "cyber" and "ICS" and "testbed" and "water". This search resulted in 72 items. After eliminating items that were not related to education in water-related ICS cybersecurity we were left with 7 strongly related literature items. This section describes the common and unique elements related to educational ICS testbed characteristics that we observed in the surveyed literature. Many of the testbeds share common general designs but implementations are varied and unique.

### 2.1 Control System Zones and Network

All reviewed works describe an ICS Network and a Enterprise Network. The ICS Network generally contains all the control technology and the I/O devices. The Enterprise Network contains the traditional IT network which has connection points into the ICS Network. Morris et. al [8], Foo et. al [4], and Yardley et. al [11] described similar ICS infrastructures. Gao et al. [5] and Čeleda et. al [9] described an ICS testbed based on the ANSI/ISA-99 four level reference model.

- Level 3: Enterprise Network
- Level 2: Supervisory Control Network
- Level 1: ICS or Control Network
- Level 0: I/O and Control Devices

The ANSI/ISA-99 reference model subdivides an ICS Network into *Level 2: Supervisory Control Networks* containing high-level automation devices, servers, and Human Machine Interfaces (HMI); *Level 1: Control Network* containing the field level control devices Programmable Logic Controllers (PLCs), Remote Terminal Units (RTUs), and Smart Relays or Intelligent Electronic Devices (IEDs); and *Level 0: Physical Control Devices* containing the field devices (actuators and sensors).

Green et al. [6] offer another variation on the Lancaster's Testbed. The Lancaster's Testbed is designed specifically for security research, which follows the Purdue Enterprise Architecture (PERA) [10]. The top down architecture describes: the *Enterprise Zone* comprised of the *Enterprise Network* and the *Site Business Planning and Logistics Network*. A *Demilitarized Zone* separates the *Manufacturing Zone* from the *Enterprise Zone*. The *Manufacturing Zone* is composed of the *Site Manufacturing Operations and Control* and the *Cell/Area Zone*. The *Cell/Area Zone* is broken into the *Area Supervisory Control*, *Basic Control*, and *Process* layers respectively. The last zone is the *Safety Zone* which connects the *Safety-Critical* controls and equipment.

### 2.2 Programmable Logic Controllers (PLC)

Most ICS testbeds contain field devices. PLCs are commonly used to implement the Master Terminal Unit (MTU) and Remote Terminal Units (RTUs). Observed units include: Control Microsystems, Inc. SCADAPack LP PLC, Allen Bradley Compact Logix L35E, Siemens S7-300, Siemens S7-1200, and a variety of trainer PLCs from DirectLOGIC, Allen Bradley, and Mitsubishi. PLCs may be programmed in a variety of ways, with the most common being Ladder Logic as described by Morris et. al [8], Foo et. al [4], and Green et al. [6].

### 2.3 Human Machine Interface (HMI)

The HMI allows students to monitor the changes in the ICS during the labs. An HMI was implemented in all of the testbeds reviewed. Morris et. al [8] was the only paper to reference a dedicated HMI device.

### 2.4 ICS Networking and Control Protocols

The reviewed literature referenced a variety of ICS networking protocols that were implemented in the educational ICS testbeds. The most commonly referenced protocol was MODBUS followed by Distributed Network Protocol (DNP3). Both MODBUS and DNP3 are used for communication between PLCs, HMI, SCADA, and other ICS equipment, and based on the lack of security in their design, both are vulnerable to tampering and attack if malicious

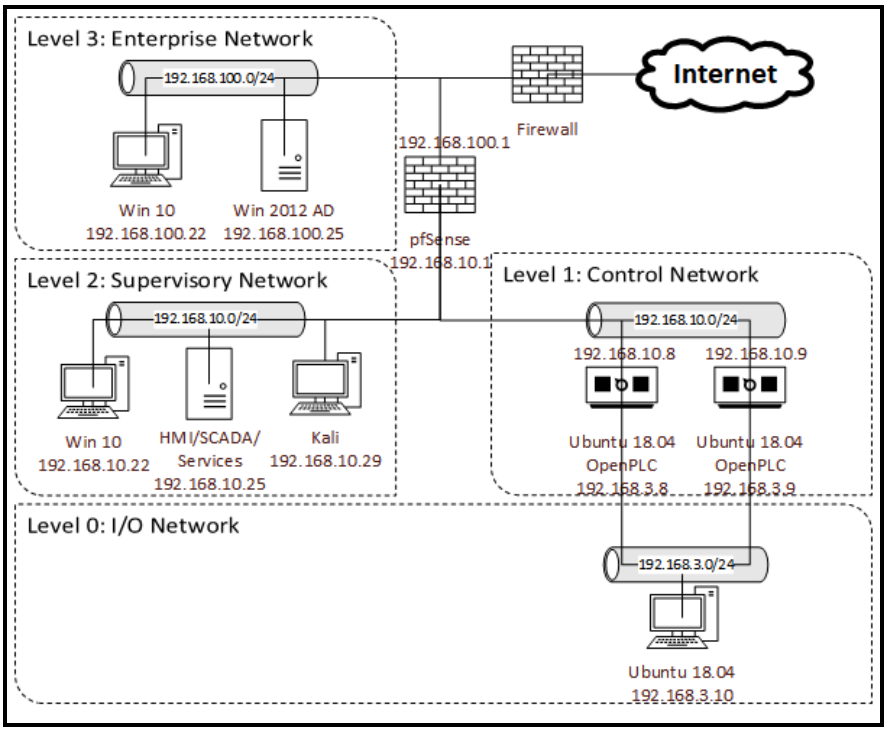



Figure 1: vWaterLabs Virtual Testbed Network Diagram

actors obtain access to the Control Network, whether directly or indirectly.

### 2.5 Virtualization

Foo et al. [4], Yardley et al. [11], Gao et al. [5], and Green et al. [6] all included some implementation of virtualization. Virtualization allows for easy setup, lab replication, and the synchronous participation from multiple students and instructors. Virtualization also facilitates Red-Blue Team exercises. The most common virtualization platform contains a single deployed ESXi Server running vSphere. This ESXi Server allows the students to connect with a laptop to the ICS Testbed.

## 3 vWaterLabs Virtual Testbed Architecture

The vWaterLabs virtual testbed is based on a simplified water treatment process. The testbed provides a low cost educational platform that is capable

of running a variety of ICS cybersecurity training scenarios. It allows semi-configurable *Enterprise* and *Supervisory* networks. This configuration allows virtual components to be added or removed based on desired configuration for a variety of educational scenarios.

### 3.1 Design

vWaterLabs testbed implementation and network are illustrated by Figure 1. The virtual testbed was implemented with four distinct zones based on the ANSI/ISA-99 reference model. Based on the literature review, network levels 1-3 are implemented as Ethernet networks. The network map in Figure 1 illustrates the Ethernet connections between the different networks.

### 3.2 Computing Infrastructure

The vWaterLabs testbed requires computing hardware to run. vWaterLabs virtual machines (VMs) may be deployed in any cloud-based or local virtualization platfrom of choice, for example vSphere/ESXi or Google Cloud; the vWaterLabs testbed has been tested on both.

The implementation of the vWaterLabs testbed requires different virtual machine (VM) configurations for each level of the ANSI/ISA-99 reference model. The following subsections explain the rationale and implementation details for each virtual machine.

#### VMs Level 3: Windows 10 and Windows 2012 Active Directory Server

Windows 10 was chosen because of its popularity in enterprise environments, and Windows 10 is the operating system that most students will likely have on their own personal computers or laptops.

Windows Active Directory (AD) and Group Policy (GP) are commonly used to authenticate users and enforce the security policies for enterprise networks. A Windows 2012 AD VM is running Active Directory and Domain Name services. This acts as the enterprise domain controller for the vWaterLabs testbed.

The AD Server contains two AD accounts (Enterprise user and Engineer user). The Enterprise user account allows authentication and access to the Win 2012 AD machine. The Win 10 VM emulates a standard workstation for an engineer user. The engineer user authenticates from the Win 10 VM via the Win 2012 AD VM and the Active Directory server.

### VMs Level 2: Win 10, HMI/SCADA/Services and Kali

As previously stated Windows 10 was selected based on its popularity both for client environments and also ICS environments.

The Supervisory Control and Data Acquisition (SCADA) machine was implemented using the Open Source software *ScadaBR*. This software is free, open source, and it contains a rich feature set that includes the Human Machine Interface (HMI), event and alert generation, and a historian application for storing data at set intervals.

The Kali machine is used for the MODBUS master simulator. As stated previously, the most common protocol for ICS testbeds is MODBUS. The Kali machine simulates MODBUS traffic via the free open source software library *modpoll*.

### VMs Level 1: Ubuntu 18.04 OpenPLC

OpenPLC on a Ubuntu 18.04 machine was used for simulating PLC devices. It features three software components including:

- The runtime software, the actual PLC software that can be run as a soft-PLC. This software can execute PLC programs in the form of Ladder Diagram (LD), Instruction List (IL), Function Block Diagram (FBD), Sequential Function Chart (SFC), and Structured Text (SL).
- The editor software, allows for PLC programming.
- HMI software, allows for HMI programming using ScadaBR software.

PLCs are created and programmed with the OpenPLC runtime software using the editor software interface. The PLCs are considered empty when the vWaterLabs testbed is deployed.

The Ubuntu 18.04 machines contain a dual network interface, because the input and output behavior is simulated on the Level 0: I/O Network. One network interface connects to Level 0 and the other connects to Level 1.

### VM Level 0: Ubuntu 18.04 Server

Hardware-in-the-loop (HIL) is a technique to accurately simulate ICS physical processes. HIL was frequently referenced by the literature.

In vWaterLabs, the HIL implementation of the physical components, including I/O, are simulated using Python scripts and the PyModbus library. The Ubuntu 18.04 Server VM operates via a Python script, which uses PyModbus, to communicate with the PLCs and update the PLC's memory values. Updating the memory values simulates physical I/O.

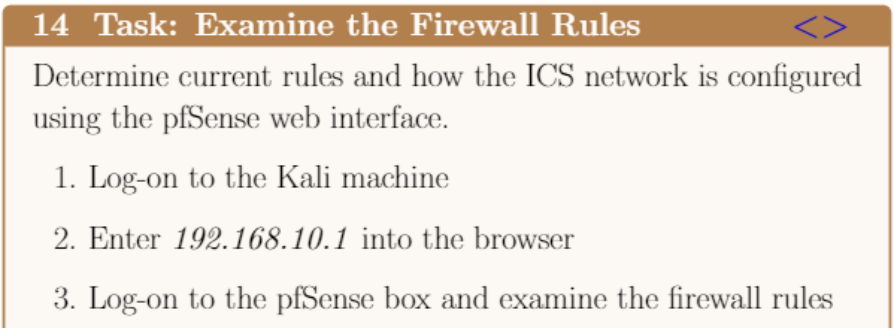



Figure 2: vWaterLabs Hands-on Vulnerability Assessment Lab Slide: Examine Firewall Rules Task

**VM: pfSense Firewall**

pfSense was selected as the firewall based on its popularity, functionality, ease of use, and free open source availability. pfSense can be configured using a graphical web interface, allowing easy access for students. pfSense supports a number of very useful options including: Virtual Private Network (VPN) Server, Domain Name System (DNS)/Dynamic Host Configuration Protocol (DHCP), Intrusion Detection System (IDS), Routing, Stateful Packet Inspection, Two Factor Authentication, and more.

This dual network interface VM works as a router and firewall between the ICS and Enterprise Network. The firewall is initially configured to allow any communication between the two networks. While not realistic, this configuration gives students an opportunity to practice configuring firewalls for the Enterprise and ICS network perimeters.

## 4 vWaterLabs Hands-On Lab: ICS Vuln. Assessment

One of the vWaterLab labs introduces students to ICS vulnerability assessments. This Lab's learning objectives are:

- Reinforce understanding of PLC, HMI, and SCADA components;
- Review details of the MODBUS protocol;
- Examine ICS network traffic using Wireshark;
- Perform an ICS vulnerability assessment;
- Implement ICS network defense techniques using a firewall.

Hands-on lab activities include: (1) introduction to assessment methods; (2) vulnerability assessment through an assessment exercise; (3) implementa-

tion of network segmentation at layer 3 using firewalls. The lab write up is a 20 page PDF booklet with the following details for each section: informational reading, estimated duration time, a task with detailed instructions, or a challenge with clues but no instructions. Figure 2 displays a sample slide. The lab can be completed in approximately two hours. For more information about the structure of these tutorial-style hands-on labs we refer the reader to two prior published works [7, 2].

## 5 Conclusion and Future Work

vWaterLabs is a virtual testbed with accompanying hands-on laboratories for Industrial Control Systems Cybersecurity education focused on water systems. In this article, we described: (1) analysis of the characteristics of educational and water focused ICS testbeds; (2) design details of a virtual-only and cost effective testbed for ICS cybersecurity education focused on water systems; and (3) introduction of an ICS vulnerability assessment lab implemented using the vWaterLabs virtual testbed. The vWaterLabs specifications, VMs, and labs are freely available at https://github.com/ICSSecurityLabs Future work will include developing a semi-automated process to create and configure the testbed environment similar to the ADLES system [1].

## Acknowledgments

This work and the used computing infrastructure were partially funded by an Idaho IGEM grant (IGEM17-001), the U.S. National Science Foundation (NSF) CyberCorps® award 1565572, and the M.J. Murdock Foundation. The opinions expressed in this article are not those of the NSF, the M.J. Murdock Foundation, or the State of Idaho.

## References

[1] Daniel Conte de Leon, Christopher E. Goes, Michael A. Haney, and Axel W. Krings. Adles: Specifying, deploying, and sharing hands-on cyber-exercises. *Computers and Security*, 74(May 2018):12–40, 2018.

[2] Daniel Conte de Leon, Ananth A. Jillepalli, Victor J. House, Jim Alves-Foss, and Frederick T. Sheldon. Tutorials and laboratory for hands-on os cybersecurity instruction. *Journal of Computing Sciences in Colleges*, 34(1):242–254, October 2018.

[3] Dragos, Inc. Threat proliferation in ics cybersecurity: Xenotime now targeting electric sector, in addition to oil and gas. Online, June 2019.

[4] Ernest Foo, Mark Branagan, and Thomas Morris. A proposed australian industrial control system security curriculum. In *Proceedings of the Hawai'i International Conference on System Sciences 2013*, pages 1754–1762. IEEE, 2013.

[5] Haihui Gao, Yong Peng, Kebin Jia, Zhonghua Dai, and Ting Wang. The design of ICS testbed based on emulation, physical, and simulation (EPS-ICS Testbed). *Proceedings - 2013 9th International Conference on Intelligent Information Hiding and Multimedia Signal Processing, IIH-MSP 2013*, pages 420–423, 2013.

[6] Benjamin Green, Anhtuan Le, Rob Antrobus, Utz Roedig, David Hutchison, and Awais Rashid. Pains, Gains and PLCs: Ten Lessons from Building an Industrial Control Systems Testbed for Security Research. *10th USENIX Workshop on Cyber Security Experimentation and Test (CSET '17)*, pages 1–8, 2017.

[7] Ananth A. Jillepalli, Daniel Conte de Leon, and Frederick T. Sheldon. CERES NetSec: hands-on network security tutorials. *Journal of Computing Sciences in Colleges*, 33(5):88–96, May 2018.

[8] Thomas Morris, Anurag Srivastava, Bradley Reaves, Wei Gao, Kalyan Pavurapu, and Ram Reddi. A control system testbed to validate critical infrastructure protection concepts. *International Journal of Critical Infrastructure Protection*, 4(2):88–103, 2011.

[9] Pavel Čeleda, Jan Vykopal, Valdemar Švábenský, and Karel Slavíček. Kypo4industry: A testbed for teaching cybersecurity of industrial control systems. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, SIGCSE '20, pages 1026–1032, New York, NY, USA, 2020. Association for Computing Machinery.

[10] Theodore J. Williams. The purdue enterprise reference architecture. *Computers in Industry*, 24(2-3):141–158, sep 1994.

[11] Tim Yardley, Suleyman Uludag, Klara Nahrstedt, and Pete Sauer. Developing a smart grid cybersecurity education platform and a preliminary assessment of its first application. In *Proceedings of the Frontiers in Education Conference 2014*, pages 1–9. IEEE, February 2014.

# Short Courses in Computer Science*

*Christopher Healy, Andrea Tartaro, and Bryan Catron*
*Department of Computer Science*
*Furman University*
*Greenville, SC 29613*
{chris.healy,andrea.tartaro,bryan.catron}@furman.edu

**Abstract**

Our institution has adopted a Maymester, a brief academic term held at the end of the academic year. Students who enroll in this optional term take a single immersive course for three weeks. This paper describes the authors' experiences in teaching five different computer science courses during the Maymester. The small class size, brief duration, and intense schedule pose special teaching opportunities and challenges. Some classes are programming intensive, while others have a broader or more liberal arts focus. Daily class activities are highly interactive. The Maymester has allowed us to offer a greater variety of new courses in subjects that otherwise could not be taught at our small college.

## 1 Introduction

In 2008, our college modified the academic calendar to include a three-week term at the end of the school year, a so-called "May Experience" or "Maymester." The purpose of this short term is to allow the faculty to offer a greater variety of elective courses, especially in subjects where there might not be enough material to justify a full semester offering. A May course could be a pilot project for a future full semester course, possibly for the major or for general education.

This paper describes the authors' experiences in teaching five different three-week courses in computer science over the last decade. Teaching such a short course introduces new challenges and opportunities. Each May course is designed to be a fast-paced and immersive experience, which is a completely different environment from the regular academic year.

## 2 Background

To ensure uniformity, all of our Maymester classes award two semester hours of credit, compared to four credits for a typical semester course. Faculty are free to schedule the course however they like. There is no maximum or minimum number of contact hours required, other than the three-week duration of the term itself. An example arrangement is for the class to meet about three hours per day, five days a week, for a total of 15 class meetings.

Maymester is optional for both faculty and students. Classes during this special term are smaller than those taught during the regular school year. Generally, the minimum course enrollment per class is six students. Enrollments are typically limited to 12, though faculty may increase this number. Maymester enjoys some popularity with the students at large. In a given year, up to 25 percent of the student body enrolls in a May class. Among the students who graduated in the last five years, the overall participation rate is 45 percent: 28 percent enrolled in a single May course during their college career, and another 17 percent took two or more. The participation rate among men (48 percent) is slightly higher than among women (43 percent).

To encourage students to enroll in Maymester, the tuition is free. The Maymester is financed by the aggregate tuition dollars collected during the regular school year. Faculty are paid at a rate commensurate with teaching summer school or teaching an overload class during the school year. In addition, the college offers grants of up to $2500 to finance the start-up costs incurred when teaching the course. For example, the faculty member may need to purchase specialized equipment or software.

At our institution, Maymester courses must be unique: they cannot be taught at any other time of the year. A May course may be used to fulfill a major requirement, but students must not be required to take any May class. It cannot fulfill a general education requirement. As a result, May course enrollment is completely voluntary. Students elect to take May classes in order to get ahead on credits, or to make up for credits lost due to a failure or withdrawal. The Maymester is also convenient for students who are not able to schedule a computer science course in the spring, such as athletes who need to underload during their season. In a given year, a student may only take one Maymester class. It is an immersive experience, akin to a full-time job for three weeks. Similarly, faculty may only teach one May course per year.

Due to their unique and elective nature, Maymester courses usually have no pre-requisites, in order to attract more students. In 2019, of the 52 Maymester courses offered throughout the college, only ten had a pre-requisite. Travel-study May courses have become especially popular in recent years. In 2019, 18 of the 52 courses were travel-study. Because Maymester courses are meant to be highly interactive, they are not intended for independent study classes or

online instruction. There was no Maymester in 2020 due to COVID-19.

## 3 Related Work

Burge and Brinkman [5] discuss computer science course design in the context of a one-course-at-a-time (OCAAT) academic calendar used at a small number of colleges. They identify some advantages and disadvantages of such a scheme. For example, students do not need to multitask between multiple courses. An instructor knows that if a student is struggling, it cannot be the result of the student spending time on other classes. The authors of the study concede the necessity for students not to stay on schedule, and the constant time pressure causing work to be rushed. It may not be the best way to teach every course.

The intense daily pace of a Maymester course is similar to that seen in OCAAT. The main difference is that an OCAAT course is intended to be equivalent to a full semester course, while for us a Maymester course carries half the credit of a typical semester course. Our May classes are electives and are never pre-requisites to other courses. OCAAT courses are also slightly longer, typically having 18 class meetings [2], compared to 15 for the Maymester.

On the other hand, some colleges feature a short term in the middle of the academic year, such as in January, as part of a "4-1-4" calendar. At these institutions, the January term is part of the regular academic year, and students must take and pass such courses, for example [1]. But at our college, the Maymester is optional. A course offered during a January term may carry equal weight to a single semester course, and lasts four weeks instead of three.

Meeker [7] describes the development of a Maymester computer science course at a liberal arts college. It was designed as a follow-on to a computer science course and a physics course, both of which were pre-requisites. At that institution, all Maymester courses are travel-study. By contrast, at our institution, most May courses are taught on campus and have no pre-requisites.

## 4 The Courses

Since the inception of the Maymester, our department has taught eleven May classes in five different subjects. The total enrollment of these courses was 85. In this section, we briefly describe the five different Maymester courses in computer science. The first three are programming courses. Their subject matter is not much different from the traditional computer science curriculum, and students would be expected to have taken at least one programming course already. The last two courses below are less technical and have no pre-requisite, in order to encourage broader student participation.

### 4.1 A Second Programming Language

This course is designed for students who have already completed CS 1 and wish to learn a new programming language. It fits well with our department's teaching philosophy. We have always stressed to our majors the value of being able to program in multiple languages. Java was the main teaching language in our department from 2001 to 2015. However, many students found our CS 1 Java course intimidating. We considered changing our introductory language to Python, a language that had never been taught before or used by any course at our college. The Maymester seemed an ideal environment to experiment with teaching Python, as a dry run for a revised CS 1 course. The goal of the course was for students to acquire a CS 1 level of proficiency in the new language.

Generally, each class meeting began with a lecture approximately 45 minutes long. On two of the class meetings, the lesson was preceded by a quiz. After the lesson, the bulk of the class time was spent in the computer lab, where students were given a set of exercises that explored the new Python concepts. Students wrote Python programs to solve short problems. Homework consisted of programming assignments to be submitted in one or two class days. The last day was a final examination, consisting of two parts. The first part was a closed-book one-hour written test of short-answer questions. The second part was a two-hour open-book practical consisting of five programming problems to solve on the computer.

### 4.2 Parallel Programming

In this course, rather than introducing students to a new language, the focus was a new programming paradigm. Students were taught to write parallel C programs using the Message Passing Interface (MPI) [8]. The structure of the course was similar to learning a new language, with a brief lesson followed by a structured laboratory.

The course began by having students assemble a Raspberry Pi cluster. The class was divided into four teams. Each team was assigned eight Raspberry Pis to work with, along with other necessary equipment, including SD cards, power cords, power adapters, Ethernet cords, plus one keyboard, monitor, and mouse. It took the first three classes for the students to hook up their machines, install the operating system and MPI software. Students, already proficient in Java, learned the rudiments of C. The rest of the course was devoted to hands-on lessons on how to use the most fundamental MPI functions, such as broadcast, barrier, scatter, and gather. Having their own cluster at their disposal allowed students to experience genuine parallel programming speedup.

### 4.3 Mobile App Development

This course introduces students to programming for the mobile application environment; specifically the iPhone and Apple Watch platforms – although this could be easily changed to the Android platform. The primary content consisted of learning the Swift programming language, software development, and the app frameworks necessary for basic app development. The first weeks' classes consisted of exploration of the technologies unique to mobile platforms (touch sensing, orientation sensing, camera, sound, animation, GPS, etc.) Midterm classes were focused on each individual student going in-depth on advanced use of technologies and developing a presentation / tutorial for the other students. The last portion of the term was spent on a final, small team project that required the students to combine multiple technologies into a new (unique) application of their choosing.

Initial class days were approximately 2-1/2 hours focused on actively exploring basic content areas in a collaborative lab, demonstration style. Later classes tended to be more independent work as they explored their selected technologies and their final project. It was noted that most students required some additional time outside of class for completion. The most successful students, aside from being self-motivated, had a well-developed idea for a project; they were highly motivated to do something "cool" (The instructor occasionally needed to temper expectations to match the limited timeframe.)

### 4.4 Science Fiction

This course focuses on works of science fiction in which computers or robots are an essential plot element. Due to the short duration of the course, the aim was to expose the students to as many examples of fiction as feasible. Thus, rather than having students read novels, they were assigned works that would take less time to digest. The repertoire included a total of 35 stories, comprising nine feature-length movies, nine one-hour television programs, three printed short stories, and fourteen radio programs. The stories were published from 1944 to 2014. The stories generally fell into two categories. The first set of stories focused on popular fears of technology, such as computers taking over the world. The second set of stories treated the subject of how well a robot can approximate human behavior.

Each day, outside of class, students were assigned a radio show to listen to, usually 30 minutes in length, or a short story to read. Fortunately, the radio shows and short stories are in the public domain, so that the students did not need to purchase anything for the course. The assigned story would be discussed at the start of the next class. The class would then have a movie or television viewing. Students were given several minutes to collect their thoughts, write

down their impressions, and share their notes with the student sitting next to them. The remaining hour of class was spent in discussion of the themes of the film, and comparisons among different works. There were no quizzes or exams during the course. Homework consisted of two 1000-word papers, plus a study guide identifying issues raised by each story.

### 4.5 e-Arts and Crafts

In the e-Arts and Crafts class, students researched interactive projects that artists, designers, crafters, and makers have created using technology while creating their own art or craft project using the Arduino LilyPad. This class draws inspiration from Buechley and colleagues [4], whose work seeks to increase participation of women in computing by asking, "how can we integrate computer science with activities and communities that girls and women are already engaged in?". Each morning, students presented research or other approved articles related to interactive and electronic arts or crafts. In the afternoon, students worked in groups on LilyPad tutorials which they developed into their own unique projects. No computer programming experience was required for this course, since LilyPad is designed to be used by novices. The course was offered twice – the first offering included both computer science and non-computer science students, while the second offering included only non-computer scientists.

## 5 Lessons Learned

Table 1 summarizes the enrollment statistics of the eleven course sections we have offered since 2009. Each year on average, there is only enough demand to support one May course in our department. Our collection of Maymester classes has attracted mainly computer science majors. They have comprised 69 of the 85 (81 percent) of our enrollments overall. We also note that few women have taken a Maymester CS course. Women have comprised just 12 of the 85 (14 percent) of our May enrollment. These figures compare to 74 percent majors and 23 percent women taking computer science classes for a regular term such as fall 2019. Although the Mobile apps course was the most popular overall, the e-Arts and Crafts course was most successful at attracting women and non-computer science majors. Given that a student may only take a single May class each year, the classes we offer are in competition with those of other departments. Furthermore, relatively few students take more than one May course during their entire college career, since other equally desirable options exist, such as getting a summer job or research assistantship.

Maymester instructors indeed find the schedule exciting yet relentless, with little room for error or falling behind. Like the mythical man-month [3], even

Table 1: Enrollment statistics of Maymester classes, 2009-2019

| Course | Iterations | Total enrollment | CS majors | Women |
|---|---|---|---|---|
| Python | 2 | 15 | 13 | 0 |
| Parallel | 2 | 14 | 14 | 0 |
| Mobile apps | 4 | 34 | 34 | 5 |
| Sci-Fi | 1 | 7 | 4 | 1 |
| Art/crafts | 2 | 15 | 4 | 6 |
| Total | 11 | 85 | 69 | 12 |

though the number of contact hours for a May class approaches that of a semester course, only so much work can be accomplished in three weeks. It is not the same as a course that meets once a week for 15 weeks, where the students have the maturation time to let ideas "sink in." In Maymester the stress level reaches its height on the last two days, when some students seem to adopt a "just get it done" mentality, and have little patience for crafting elegant solutions.

We have found that all of the students were able to immerse themselves into the class. Teams of two worked well in classes involving very detailed tasks, such as building the Raspberry Pi cluster. In the e-Arts and Crafts class, we experimented with teams of three. In that class it was helpful if one member of each team already had programming experience. But larger teams are not feasible due to the low enrollment. Having students working together helps everyone keep to the daily schedule. The instructor needs to pay attention to how much the students accomplish each day, in order to readjust the next day's schedule as needed.

When preparing a course, we found it essential to have each day carefully planned well in advance, but with flexibility to allow for hiccups. Technical problems consumed precious time. For example, once we discovered that we did not order enough hardware components, and had to have additional ones delivered overnight at considerable expense. Multi-day projects should be divided into well-defined portions with logical stopping points each day. When a lab begins, it is a good idea to let the students know how far they should get so they can pace themselves.

All of the students who enrolled in a computer science Maymester course successfully completed it. Class absences were practically nonexistent. But unfortunately, across the entire college many students see the Maymester primarily as an opportunity to improve their grade average. Over the last ten years, 84 percent of Maymester grades have been A's, compared to 43 percent during a regular semester. Therefore, students may expect to receive an A by default.

In our course evaluations, one student who received a B commented that "the grading was unreasonably harsh."

Already the Maymester has had a lasting effect on the curriculum of our regular academic year. After two successful iterations of the Python course, our department decided to change the CS 1 teaching language from Java to Python in 2015.

## 6 Applications to Full-Term Courses

Some of the pedagogical techniques employed in the Maymester can be adapted to regular semester-long courses as well. A Maymester course, such as the ones described earlier in this paper, could become one component of a semester course. In an upper-level elective course, it may be desirable for students to immerse themselves in a specific topic for a short period of time. For example, a mobile phone app development project could be a prototyping case study for a semester course in human-computer interaction. And the experience of developing a Raspberry Pi cluster can be incorporated into a general course in computer organization.

Regular semester courses lack the full-day immersion that a Maymester course offers. But the applied learning practices can be utilized in any class. May courses favor a flipped classroom format [6] over traditional lectures. First, content is presented to students in advance, such as a video tutorial. Second, class time is devoted mainly to hands-on laboratory work in small groups. Students are responsible for consulting hardware and API documentation relevant to their needs. Finally, each group can report to the rest of the class what they discovered or accomplished.

## 7 Future Possibilities

In addition to the five courses mentioned earlier, we are planning to offer at least four more Maymester courses in computer science. One such course is on research methods, analogous to similar courses that are taught in biology, chemistry, and psychology. Many undergraduate students in our department conduct summer research projects under the direction of faculty. Such a course would introduce students to background concepts and skills that are common to all computer science projects, such as ethics, data and code management, and presentation skills.

Secondly, we have prepared a new course on remote sensing. This is a course where students create a system that communicates inside a natural environment, such as a field or forest. We plan to use the Arduino for students to create interactive projects. Arduino is an electronics prototyping platform, which

includes a circuit board, LED lights, sensors, and a software development environment. It is simple enough to use that students could take this course without a pre-requisite. The specialized hardware costs associated with this course are higher than the other Maymester courses (about $200 per student), so additional funding would be necessary to offer this course.

The Maymester can make it easier for computer science students to complete a study-abroad experience. A traditional study-abroad program lasts a whole semester, and students take courses from two or more departments, built around a common theme. It is usually difficult to find a logical fit for computer science courses into a semester-long study-abroad program. But a study-abroad program during a Maymester can focus on a single course, without having to find extra-departmental courses to combine with. We know there is demand for study-abroad programs in computer science. In 2019, eleven of our own majors enrolled in a Maymester study-abroad course offered by other departments. Therefore, we plan to offer a Maymester course on the history of technology and codebreaking in the UK. This is already a component of an existing semester-long study-abroad course [9]. Highlights of this journey include the Belfast shipyards, the first limelights used in theaters, and most notably the huts at Bletchley Park where Alan Turing built the first computer.

Finally, recognizing that game simulations are a part of most of our programming classes, another future course will analyze board games. This course is intended to be interdisciplinary, and team-taught with faculty from both computer science and mathematics. The course will focus on how game boards are modelled computationally as a data structure, either as a tree or an array. In addition, students will see how the statistical properties of random numbers used in games can affect player choices and the outcome of the game.

## 8 Conclusion

Over the last decade, our department has offered five different computer science courses taught during the three-week annual May Experience. These courses are free electives for the students and feature unique benefits. They give the faculty an opportunity to experiment with new elective courses, especially courses that do not require an entire semester of time for the students to master. For the students, it is an immersive and collaborative experience, with significantly smaller class sizes and much more attention from the instructor.

Not every course can fit into a three-week term. Creating and preparing the course requires the instructor to devise a set of activities that all students can realistically achieve in just three weeks working full time. Technology usually advances faster than the curriculum. In the Maymester we can offer new courses in subjects that otherwise could not be taught at our small institution.

Teaching Maymester classes has significantly broadened the variety of courses our department has offered in recent years.

## References


[1] *2015-2016 Catalog.* Wofford College.

[2] Cornell college faculty handbook 2019-20. https://www.cornellcollege.edu/academic-affairs/faculty-handbook/2019-20\%20Faculty-Handbook-Revised\%2002-28-2020.pdf.

[3] Frederick Brooks. *The Mythical Man-month: Essays on Software Engineering.* Addison-Wesley, 1995.

[4] Leah Buechley, Mike Eisenberg, Jaime Catchen, and Ali Crockett. The lilypad arduino: using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 423–432, New York, NY, USA, 2008. ACM.

[5] Janet Burge and Bo Brinkman. Teaching and learning under pressure: Intensive (accelerated, block) computer science courses. In *Proceedings of the ACM SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '17, page 721, New York, NY, USA, 2017. ACM.

[6] Mary Lou Maher, Celine Latulipe, Heather Lipford, and Audrey Rorrer. Flipped classroom strategies for cs education. In *Proceedings of the ACM SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '15, pages 218–223, New York, NY, USA, 2015. ACM.

[7] Paige Meeker. Interdisciplinary travel courses – creating 'magical' experiences. *Journal of Computing Sciences in Colleges*, 29(4):43–49, 2014.

[8] Peter Pacheco. *An Introduction to Parallel Programming.* Morgan Kauffman, 2011.

[9] Kevin Treu. History of technology and discovery: a study away experience in computer science. *Journal of Computing Sciences in Colleges*, 36(5):160–167, 2021.

# A Web-Based Toolkit for Exploring Cryptography*

*Mikel Gjergji, Edmund A. Lamagna*
*Department of Computer Science and Statistics*
*University of Rhode Island*
*Kingston, RI 02881*
*mikel_gjergji@uri.edu, eal@cs.uri.edu*

**Abstract**

A set of web-based tools has been developed to assist in teaching and learning cryptography. The website provides a uniform environment for exploring topics commonly taught in introductory courses. These include substitution and transposition ciphers, block ciphers (DES), public-key infrastructure and encryption (Diffie-Hellman key agreement, RSA), and hashing. There are also tools for investigating number theoretic concepts such as modular arithmetic and the Euclidean algorithm. The applets allow users to explore visually how the methods operate. An instructor can use the tools in the classroom to explain the algorithms and present examples. Students use the site to explore methods on their own, solve problems, and crack cryptographic challenges. The tools eliminate the need for students to write programs to perform these computational tasks, enabling them to focus on important algorithmic and mathematical ideas.

## 1 Introduction

Cryptography is an important real-world application of computer science and mathematics. The subject can be taught to student audiences at all levels. For liberal arts majors or general education students, a course can be constructed at an introductory, pre-calculus level. A successful course for this audience provides an introduction to mathematical and computational thinking, exposure

to a variety of contemporary mathematical topics, and a discussion of societal issues relating to security and privacy. An upper-level course on cryptography is an appropriate elective for undergraduate majors in computer science, mathematics, and engineering. For these students, cryptography provides an important computational application of mathematics allowing them to integrate a variety of concepts, some of which should be familiar and others new.

The subject of cryptography builds on a wide range of mathematical areas. The notion of functions, their inverses, and functional composition are central to any mathematical treatment of cryptography. Some classical codes like the Spartan scytale and transposition ciphers are based on permutations. Modular arithmetic provides the foundation for classical codes like shift and affine ciphers, as well as contemporary public-key methods. Probability and statistical techniques can be used to crack many classical codes including polyalphabetic substitutions like the Vigenère cipher and the Enigma machine. Matrices provide the basis for Hill ciphers and can be used in the description of other cryptographic schemes. Boolean algebra is used to describe the linear feedback shift registers employed in scrambling communication signals, the Data Encryption Standard (DES), and the hash functions used to verify digital messages. The Advanced Encryption Standard (AES), the successor to DES, is based on finite fields. Topics from number theory (Euclid's algorithm, Fermat's Little Theorem, factoring, discrete logarithms) form the basis for understanding the RSA and ElGamal public-key cryptosystems. Finally, the security of modern cryptosystems is assessed in terms of computational complexity using the techniques of algorithm analysis.

A set of web-based tools has been designed and implemented to assist in teaching and learning cryptography. The tools are assembled in a uniform environment for exploring topics typically covered in an introductory course. The website includes applets for encryption, decryption, and cryptanalysis. Importantly, tutorial material about the techniques is also provided. The applets allow users to trace visually, in a step-by-step manner, how the methods operate. There are applets for classical substitution and transposition ciphers, a simplified version of a block cipher (DES), Diffie-Hellman key agreement, public key encryption (RSA), and hashing. Also included are tools to explore underlying concepts from number theory such as modular arithmetic, modular exponentiation with the binary method and Fermat's Little Theorem, and greatest common divisor computation with the Euclidean and extended Euclidean algorithms. There is also a version of Summerside Makerspace's Universal Enigma Simulator[5]. In addition, computational support is provided to help perform cryptanalytic attacks on classical ciphers and small instances of public-key ciphers. Instructors can use the tools in the classroom to explain the algorithms and present examples. Students use the site to explore

the methods on their own, to solve problems, and to crack cryptographic challenges. The tools eliminate the need for students to write programs to perform these computational tasks, enabling them to focus on important algorithmic and mathematical ideas. At our institution, the tools have been used successfully in conjunction with a freshman honors course for non-majors and an upper-level course primarily for computer science majors. The materials are textbook-independent and texts that have been adopted for these courses include Barr[1], Beutelspacher[2], Cozzens and Miller[3], Holden[4], and Trappe and Washington[7].

## 2 Affine ciphers

The first non-trivial cipher usually encountered in an introductory cryptography course is the affine cipher. This method provides a good springboard for discussing modular arithmetic, inverse functions, and greatest common divisors (GCDs) and their computation.

To facilitate a mathematical treatment of cryptography, textbooks adopt the convention of encoding letters as numbers, with the most common scheme being `A` = 0, `B` = 1, `C` = 2, ... , `Z` = 25. An affine cipher encrypts a plaintext letter $x$ to the ciphertext letter $y$ using the linear transformation $y = ax + b \bmod 26$. The key is a pair of numbers $(a, b)$ in the range 0 to 25, where $a$ is the multiplier and $b$ is a shift amount. In order to decrypt, the multiplier $a$ must have an inverse $a^{-1} \bmod 26$; i.e., $\mathrm{GCD}(a, 26) = 1$ and $a \cdot a^{-1} \equiv 1 \bmod 26$.

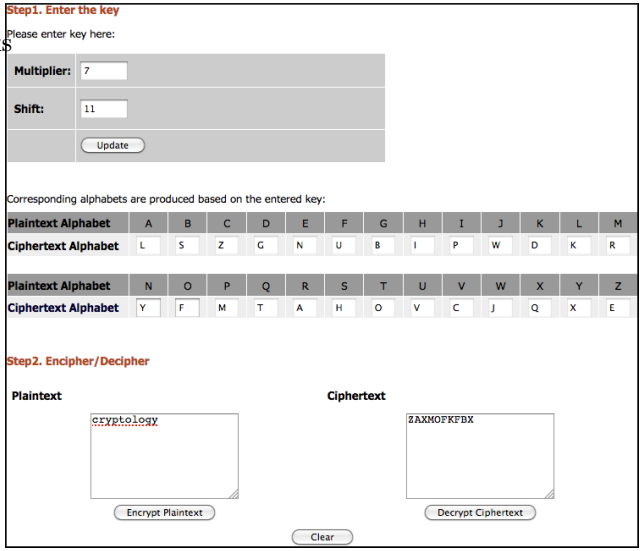



Figure 1: Affine cipher tool

The tool for encrypting and decrypting affine ciphers is depicted in Fig-

ure 1. After entering the key, the resulting ciphertext alphabet is shown below the corresponding plaintext letters. By way of example, the plaintext letter `c` encodes to 2 and $y = ax + b = 7 \cdot 2 + 11 = 25$ gives the ciphertext letter `Z`. The encryption of the word `cryptology` with key (7, 11) is shown in the figure. A good exercise for students is to determine the decryption function, $x = a^{-1}(y - b) \bmod 26$, then enter the corresponding multiplier and shift amounts in the tool, and observe the alphabet used to decrypt a message. For this example, $x = 15y + 17 \bmod 26$.

## 3 Vigenère cipher cryptanalysis

Monoalphabetic substitutions like the affine cipher easily succumb to letter frequency attacks. In English, for example, the most commonly occurring letter is `E`, followed by the letters `T` and `A`; `X`, `J`, `Q`, `Z` seldom occur. Polyalphabetic ciphers like the Vigenère cipher were developed to circumvent such frequency attacks.

An example considered in elementary cryptography courses is the Vigenère cipher, which is just a sequence of shift ciphers. The ciphertext is found by adding the numeric equivalents of the plaintext to those of the key modulo 26.

| | |
|---|---|
| `Maryhadalittlelamb` | plaintext |
| `POEMPOEMPOEMPOEMPO` | key |
| `BOVKWOHMAWXFASPMBP` | ciphertext |

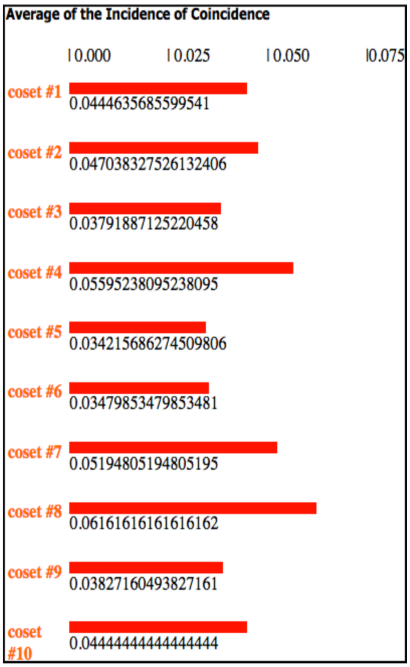
For example, `M` (12) + `P` (15) = `B` ($27 \equiv 1 \bmod 26$). The longer the keyword, the better the letter frequencies of the ciphertext tend to be masked.

Cracking a Vigenère cipher is a two step process: (1) determining the length of the keyword, and (2) determining the keyword itself by examining the frequencies of letters in each coset. Ciphertext produced by a 4-letter keyword has four cosets (one generated by each letter in the keyword). Both steps of the cryptanalysis can best be explained through the notion of the index of coincidence, $I$. This index provides a measure of how random the letter frequencies are in a sample of text. If all 26 letters occur with equal probability, the index is $I_{\text{random}} = 1/26 \approx 0.0385$. Due to the letter frequencies of English, however, passages of text typically exhibit an index of coincidence that is much higher, $I_{\text{English}} \approx 0.065$.

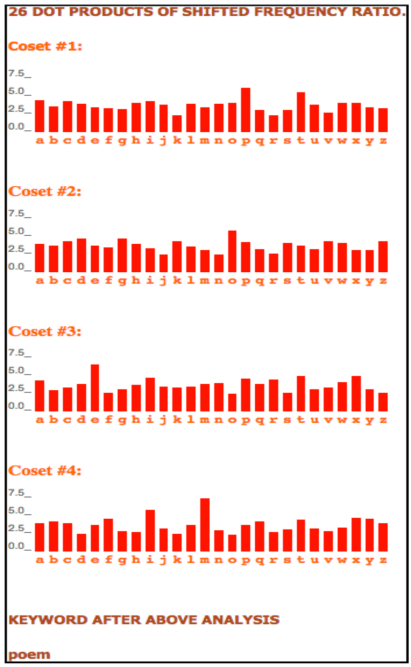
Figures 2a and 2b illustrate a portion of the cryptanalysis of the first verse of the nursery rhyme, "Mary Had a Little Lamb." The process begins by reporting a count of the letter frequencies in the ciphertext, along with its index of coincidence and an estimate of the keyword length based on the index (not shown). Although the estimate is good (4.26) for this example, the method is often highly inaccurate. A more precise way to estimate the keyword length is shown in Figure 2a. The ciphertext is divided into cosets up to some maximum

number—in this case 10. The average value of $I$ over each of the cosets will be close to $I_{\text{English}}$ for the correct keyword length (or a multiple of it), and approximately $I_{\text{random}}$ for incorrect keyword lengths. This is illustrated Figure 2a, which suggests the keyword size is 4 for the example.

Based on a keyword of length 4, Figure 2b shows the calculation that produces the likely keyword, `poem`. The keyword is computed by taking dot products of a 26-component vector containing the frequencies of the letters (`A` through `Z`) in each coset with vectors containing shifts of the frequencies of the letters in English. This product should be close to $I_{\text{English}}$ for the shift corresponding to the correct key letter and about $I_{\text{random}}$ for the other shifts. (See Barr[1].) It is perhaps surprising that the keyword can be correctly determined for the rhyme from such a small sample of ciphertext!




(a) Vigenère cosets




(b) Vigenère dot products

Figure 2: Output from the Vigenère tool

## 4 Block Ciphers and SDES

Advances in computing power have resulted in the Data Encryption Standard (DES) being superseded by the Advanced Encryption Standard (AES). Unfortunately, an understanding of AES requires some knowledge of finite fields. A simplified version of DES still provides an excellent vehicle for introducing students to computer-based block ciphers.

Trappe and Washington[7] present a simplified version of DES, called SDES, with almost all the features of DES. The toolkit includes an emulator that depicts visually, in step-by-step fashion, a single round of SDES (see Figure 3). The algorithm operates on 12-bit blocks consisting of two 6-bit characters. Each round scrambles the right character of a block using 8-bits of a 9-bit key and a pair of S-boxes. The 6-bit input character is first expanded to 8, and these bits are XORed with the 8 key bits used in the current round. The result is then divided into two 4-bit halves, which are used to index two S-boxes. The first bit indicates the row, and the other 3 bits specify one of the 8 columns. The appropriate 3 bits taken from each S-box are concatenated and then XORed with the left input character of the block. The result is the right character output for this round. The left character output is simply the right character input, and it will be scrambled in the next round.

The emulator has a box where users enter a message to be enciphered. Since 6-bit characters are supported, the input can consist of lower case (`a`–`z`) and upper case (`A`–`Z`) letters, digits (`0`–`9`), periods and spaces. There are boxes to enter the 9-bit key and the number of rounds used to encipher a block.

An important pedagogic feature of the tool is that it can also be used for decryption. Ciphertext from the output box can be fed back to the input box. When the user selects "Decrypt," the keys are used in reverse order, round by round, to produce the original plaintext.

## 5 Hashing

Cryptographic hashing is a message validation and authentication technique. The goal is to take a message of arbitrary size and produce a fixed size "digital fingerprint." Ideally, it should be virtually impossible to find another message with the same hash value. Moreover, a small change in the message should produce a hash value that appears to be entirely uncorrelated to the original.

Real-world hash functions are exceedingly complex, highly non-linear bit scramblers. The hash function applet in the toolkit is by no means "industrial strength," but is intended to provide a "toy hash" that students use to get the flavor of how hash functions operate.

The tool is depicted in Figure 4. It operates on 24-bit blocks of text con-

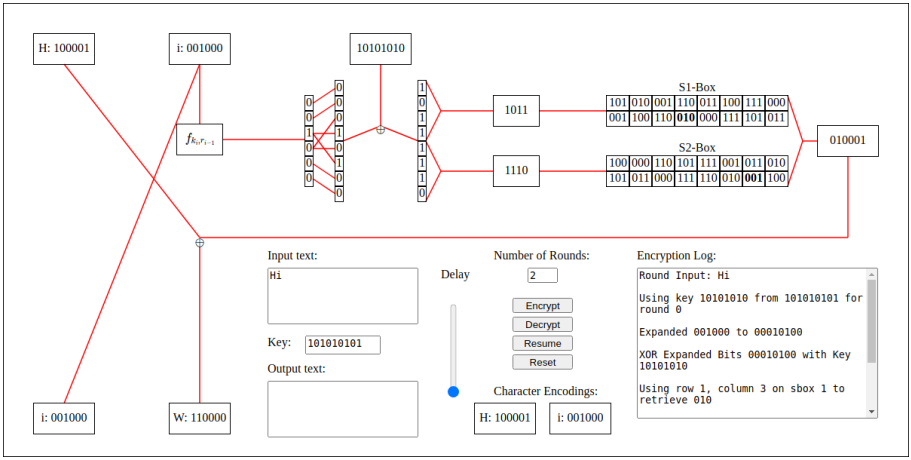



Figure 3: Simplified DES tool

sisting of four 6-bit characters, encoded the same way as the SDES tool. The top row in the figure shows the initial 24-bit value of the hash—the characters `SHA1` in the example. This row is updated after each block is processed. The second row shows the four characters and corresponding bits of the current text block. In our example, the text is `MD5`, padded with a space to give the block four characters. These two 24-bit strings are XORed and the result is permuted, giving the bits corresponding to the character string `O8vK`. (The user can customize the permutation used for each character. If it is desired to keep things simple, this feature can be turned off.)

Next, each of the four 6-bit outputs in this row is intermixed with bits from the outputs of the other three characters with another XOR. A 6-bit string is formed by taking the first two bits of the character immediately to the right (cyclically), the second two bits from the character to the right of that, and the last two bits from the character to the right of that. Each pair of bits is color coded in the applet to show where it originates. In the example, `O8vK` is XORed with `OuOT` to produce the hash value for this block, `C09j`. Utilizing bits from the other characters creates a stronger hash.

As described above, the hash tool produces a Message Digest, where a known initial value is used to verify the integrity of a message. The tool can also be used to create a Message Authentication Code (MAC) by changing the initial hash value or altering the permutations used after the first XOR. With a MAC, the receiver can verify not only the integrity of a message but also the authenticity of the sender. To do so, the initial hash value and the permutations are agreed upon in advance and kept secret by the communicating parties.

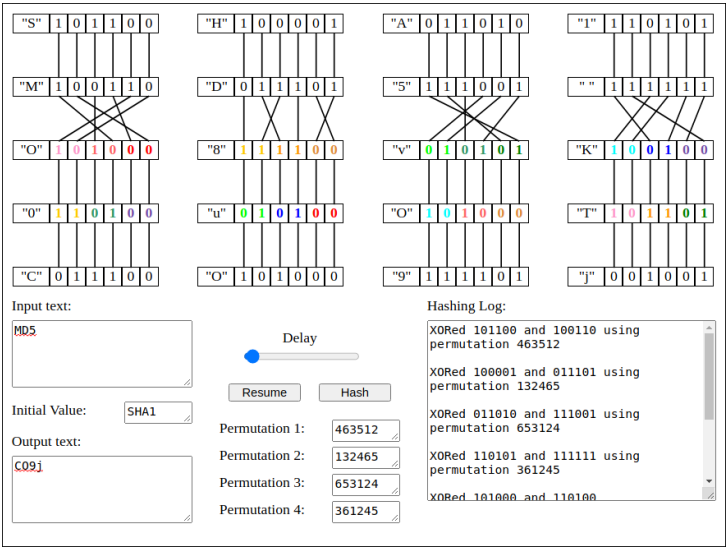



Figure 4: Hashing tool

## 6 Public-Key Cryptography and RSA

In a public-key system, each user has a pair of keys: a public key that is known to everyone, and a private key known only to the user. The public key is published and used by everyone to send encrypted messages to the user. The private key is used to decrypt messages enciphered with the public key. In order for a public-key system to be secure, it must be computationally infeasible to determine a private key knowing the corresponding public one.

The best known public-key cryptosystem is RSA[6]. Its security depends on the difficulty of factoring an integer that is the product of two large primes, a problem for which no efficient algorithm is known.

The toolkit contains an implementation of RSA for small sized primes (see Figure 5). A user begins by entering two prime numbers, $p$ and $q$. The tool then calculates and reports two moduli, $m$ and $n$, used by the algorithm: $m = pq$ and $n = (p-1)(q-1)$. Next the user enters an encryption (public) exponent $e$ that is relatively prime to $n$. The tool next uses the extended Euclidean algorithm to find the private decryption exponent $d$, which is the multiplicative inverse of $e$ mod $n$: $ed \equiv 1 \bmod n$. The applet reports $d$ and the public key, or $(e, m)$-pair. It also shows the block size $b$, or number of characters that can be enciphered with a single application of the algorithm. This limit is determined by the value of the public modulus $m$.

The user can now encode a message as a sequence of $b$-digit numbers modulo 29 (a prime). Space, period, and comma are encoded as 0, 1 and 2; the letters `A` to `Z` are encoded as 3 through 28, respectively. (0 and 1 are problematic since they are unchanged when raised to a power. This is why the encoding of `A` has been shifted from its usual position to 3.) On the next line, users can encrypt a number(s) $x$ as $y = x^e \bmod m$, or decrypt a number $y$ as $x = y^d \bmod m$. On the last line, numbers can be decoded back to text by reversing the encoding process. Slashes (`/`) are used to separate blocks. This allows use of the tool as a decryption, as well as encryption, device. Text containing slashes can be entered on the "Encode" line, and the tool deals appropriately with slashes when converting to numbers.

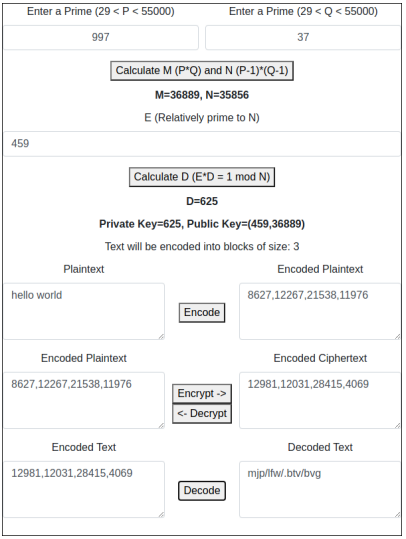



Figure 5: RSA tool

## 7 Conclusion

The website includes many other cryptographic tools that are not described due to lack of space. The authors have tried to convey the "look and feel" of the environment here. For example, the demos for classical ciphers (shift, Hill, Spartan scytale) resemble the tool for affine ciphers. The website has evolved over a decade. New tools are continually added, as old ones are updated or completely revamped. The SDES and RSA tools were originally implemented as Java applets requiring a plug-in. As this approach to web development has been deprecated, the tools were redesigned and implemented in JavaScript. In the redesign, some features were improved and new ones added. The website once had a Java applet simulating the Enigma machine. This was replaced with Summerside Makerspace's excellent Universal Enigma Simulator[5], adapted to our site. The environment is a continuing work in progress. Plans are afoot to implement a simplified version of the Advanced Encryption Standard (AES) and a tool for the ElGamal public-key cryptosystem.

The website has been used many times in offerings of a general education course for non-majors and an upper level course primarily for computer sci-

ence students. Classroom use of the tools and student feedback has informed development of the environment over time. Experience has shown that an overwhelming majority of students find the tools useful in learning cryptography and enjoy working with them. Their use makes the subject come alive far more than reading and studying about ciphers in a textbook. Students particularly enjoy solving non-trivial cryptographic challenges with the tools.

The web toolkit is accessible at: https://www.cs.uri.edu/cryptography/

## 8 Acknowledgement

This project evolved from a set of tools developed by students in a graduate cryptography course at URI over a decade ago. One student in the class, Surender Chiluka, approached the second author with the idea of improving the tools, adding new ones, and collecting them on a website with an attractive, uniform interface. Two other students, Krupesh Patel and Zach Oliveira, performed major upgrades at different times by improving the code, incorporating additional methods, and including a section of number theory tools. Finally, the authors are grateful to the many students who have used the tools in classes over the years for reporting bugs and for their helpful suggestions.

## References

[1] Thomas H. Barr. *Invitation to Cryptology.* Prentice Hall, Upper Saddle River, NJ, 2002.

[2] Albrecht Beutelspacher. *Cryptology.* Mathematical Association of America, Washington, DC, 1994.

[3] Margaret Cozzens and Steven J. Miller. *The Mathematics of Encryption.* American Mathematical Society, Providence, RI, 2013.

[4] Joshua Holden. *The Mathematics of Secrets.* Princeton University Press, Princeton, NJ, 2018.

[5] Summerside Makerspace. Universal enigma machine simulator. https://summersidemakerspace.ca/projects/enigma-machine/, 2020.

[6] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.

[7] Wade Trappe and Lawrence C. Washington. *Introduction to Cryptography with Coding Theory (2nd ed).* Prentice-Hall, Upper Saddle River, NJ, 2005.

# Students' Consistency in Computational Modeling and Their Academic Success*

*Elena Izotova, Jason Kiesling, Fred Martin*
*Department of Computer Science*
*University of Massachusetts Lowell*
*Lowell, MA 01854*
*{eizotova,jkieslin,fredm}@cs.uml.edu*

**Abstract**

In this study, an assessment was designed to measure consistency in how subjects interpreted the effect of programming statements. The assessment consisted of 24 multiple-choice items which tested student interpretation of assignment and equality operators. Answers were analyzed to determine each subject's "Consistency Score," which represents their consistency in this interpretation. The assessment was administered to computer science undergraduates at a public research university in the Northeast USA. The respondent results (n=128) were compared to the students' self-reported department GPA with the goal of determining whether consistency is correlated with student success. We found a positive correlation between a student's Consistency Score and their department GPA, with strong significance. This suggests the use of this instrument as a diagnostic for supporting students. This paper presents the design of the assessment, how the Consistency Score is calculated, and the study results.

## 1 Introduction

Student success rates in the introductory computer science courses at colleges and universities worldwide are low. Research has indicated that it is typical for about 30% of students to fail to successfully complete the first-semester course

[1]. At our institution we have similar outcomes. This problem is recognized to be long-standing and one that is independent of language taught [12].

We suggest that understanding students' computational mental modeling may provide insight and an opportunity to address this problem. Computation mental modeling describes the ways that a novice programmer justifies computer behavior and explains how code works. Some common modeling errors include believing that the computer has intuition, that all code runs simultaneously rather than in order, and that assignment empties and zeroes out one variable into another [4].

No matter if a novice programmer forms models correctly or incorrectly, their consistency may be an important indicator to support success in a computer science classroom. If a student consistently believes an incorrect model, all they may need is some guidance in the right direction. However, if a student vacillates between conflicting models, this may make programming more difficult.

This study examines the hypothesis that a student's consistency of mental modeling is correlated with their success in the academic program as measured by their self-reported department GPA, with the goal of establishing the importance of mental modeling in programming. This work is an important first step at re-examining the way that computer science educators support students. Students that are struggling in their computer science classes may need additional guidance in their computational mental modeling so that they may be more consistent.

## 2 Background

This section examines foundational literature on novice programmers, mental modeling, and programming assessments.

### 2.1 Notional Machine and Mental Modeling

As introduced by du Boulay, the notional machine is "the general properties of the machine that one is learning to control" [4]. Novice programmers often have difficulties connecting the behaviors of the physical machine and the notional machine, and consequently there are a number of common incorrect mental models that novices form about the notional machine. Pea *et al.* explore the common mistakes that programming students make as they learn about and form their own mental models about the machine [8]. These prior works touch on faulty or incomplete understanding of programming concepts, without examining other properties of a student's knowledge systems. Instead of looking at "right" or "wrong," there may be other important aspects such as stability.

Although there are many different aspects to forming mental models, we are most interested in how people solidify and become consistent in the formation process. According to Johnson-Laird, people are able to stop searching for alternative models when their models are able to form valid conclusions; alternatively, people need to modify their mental models when they no longer adequately explain input [5].

In general, novice students tend to have trouble synthesizing information and generalizing knowledge. In order to expand the modeling process, Linn suggests positive relationships between student competency and teaching code templating, reusing code blocks, time spent debugging, and generalizing algorithms [6]. Each of these activities listed help an individual learn to form their own models of the notional machine, rather than memorizing a given set related to a specific language. A typical curriculum, though, may begin with learning and familiarizing language features, as opposed to introducing generalized design and problem solving concepts. This focus may be preventing students most in need of these resources from developing confident mental modeling in new environments.

A characteristic of a skilled programmer is the ability to quickly abandon questionable assumptions, whereas a novice programmer tends to think with a more narrow scope. Robins *et al.* [9], Stefik & Siebert [11] and others suggest relationships between the choice of programming language and the success of novice programmers. Much effort has been invested in finding the "best" language to teach introductory computer science courses, but studies show that choice of language does not evidently determine student success (e.g., [12]). Perhaps a focus on student mental modeling as opposed to understanding the syntax would have impact.

### 2.2 Programming Assessment

Our curiosity of consistency of mental modeling and how it relates to academic success was seeded by prior work done by Dehnadi & Bornat [3, 2]. Their studies suggest the possibility of a programming aptitude test. They compare the results from their test to the academic success of students across multiple institutions. The programming challenges on the exam were written in Java, and consisted of one to three assignment operations.

We found it problematic to give programming challenges in an existing programming language, in the case of prior knowledge interfering with the results, and giving unfair advantages. To avoid these issues, we created our own temporary language, out of pieces from multiple different languages. According to a multi-national, multi-institutional study [13], it is possible to maintain the difficulty level of a programming assessment across different languages. Centrally, our study defines a "Consistency Score," a percent score for rate of

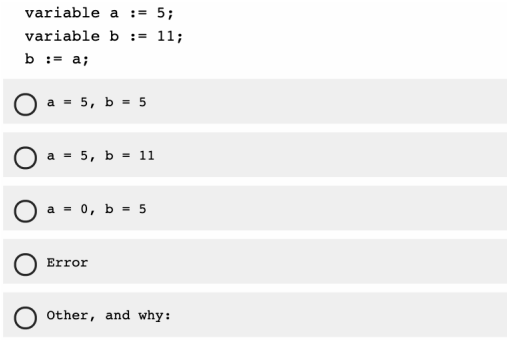


Figure 1: Assessment Item #1

consistency. This extends prior work which categorized students into the binary buckets of "consistent" or "inconsistent."

## 3 Methodology

This section covers the form of the programming assessment, the way the answers relate to mental models, and other choices made in the creation of the instrument.

### 3.1 University Makeup and Instrument Administration

The University of Massachusetts Lowell (UMass Lowell or UML) is a public research university in the Northeast United States. UMass Lowell offers more than 100 Undergraduate programs, over 40 Master's Programs, and over 30 Doctoral Degree Programs to more than 18,000 students. The assessment was distributed to all 766 undergraduate computer science majors via email. Respondents were given the incentive of a lottery to win one of ten $10 Amazon gift cards for participation.

### 3.2 The Programming Assessment

The programming assessment used a bank of 24 multiple-choice items. The items are given in an invented, single-use language with pieces from multiple different languages with some pieces completely new. The goal of the exam is to find consistency of computational mental modeling rather than determining if a subject is "correct" or "incorrect."

All subjects received the item presented in Figure 1 at the beginning of the programming assessment. This item presents two variable declaration statements (e.g., `variable a := 5;`) followed by a statement that is more open to interpretation (`b := a;`). All subjects are thus exposed at the outset to this

Table 1: Models from Assessment Item #1

| # | **Answer** | **Model** |
|---|---|---|
| 1 | a=5, b=5 | M1: := works as assignment |
| 2 | a=5, b=11 | M2: := works as Boolean equality |
| 3 | a=0, b=5 | M1, M3: := empties the right into the left |
| 4 | Error | ME: Error |

possibly new syntax, :=, and have the opportunity to be more flexible with their modeling, minimizing false inconsistency.

Subsequently, each subject received the remaining items in a randomized order, and subjects had a total of three minutes to complete the assessment. The purpose of this instrument design was to ensure that subjects would invest focused time in performing the assessment, and that each item was answered approximately the same number of times. This did result in some subjects answering more questions than others. Subjects answered between zero and 24 items.

Each multiple-choice answer follows one or more distinct computational mental models. For example, a model may be that the := operator represents assignment, while another may be that the := operator represents the Boolean equality operator. The programming challenges are of various difficulty and contain many different potential models, but each question contains either a Boolean equality operator or an assignment operator. By the end of the assessment, a subject obtains a series of percentages for each model: the number of times the subject chose a multiple-choice answer representing a particular model (the numerator) over the total number of times that model could be selected in the questions that they answered (the denominator). These percentages represent the consistency with which the subject follows each model. Ultimately, these two complementary models collected the largest quantity of data, thus they were used to determine the overall Consistency Scores for each individual.

The assessment item in Figure 1 is broken down into its model components in Table 1. Each answer represents one or more models. In this example, answer 1 represents Model 1 (M1), which is the understanding that := is an assignment operator. Answer 2 indicates Model 2 (M2): that := is a Boolean equality operator. Answer 3 indicates two models: since we can observe the behavior of an assignment operator transferring and emptying the source variable into the other (M3), we can also conclude that the user believes that the := symbol represents the assignment operator (M1). Answer 4 is a non-specific error model (ME).

Table 2: Assessment Results for Three Subjects

| # | CS GPA | M1: := Assignment | M2: := Boolean | Consistency |
|---|---|---|---|---|
| 1 | 3.75-4.0 | 19/19 = 100% | 0/19 = 0% | 100% |
| 2 | 3.5-3.74 | 19/21 = 90.48% | 2/21 = 9.52% | 90.48% |
| 3 | 2.5-2.74 | 12/19 = 63.16% | 7/19 = 36.84% | 63.16% |

## 4 Analysis

This section presents the results and statistical analysis of the data collected from distributing the programming assessment to a group of undergraduate students at UML. From a population of 766 undergraduate computer students. 198 individuals completed the survey; these were filtered to 128 students who: indicated they were current undergraduates, completed at least four items corresponding to models M1 and M2, and provided their department GPA.

### 4.1 Computation of Consistency Score

After coding each multiple-choice answer in the assessment to one or more computational mental models, we automated the scoring process by running the raw data through a program. After processing the data, we yielded a set of 128 rows, where each row represents one subject.

The computational mental models used to determine the Consistency Score for each subject were complementary, meaning that the percent consistency for one model is naturally inverse to the percent consistency of the other. As a consequence of the complementary nature, the consistency percentages range from 50% to 100%, rather than 0% to 100%. The resulting Consistency Score is the higher of the two percentages.

Table 2 shows the consistency data from three of the 128 subjects. We see a range of consistency behaviors. Subject 1 self-reported a CS GPA of 3.75–4.0, chose answers corresponding to M1 100% of the time, and chose answers corresponding to M2 0% of the time. Subject 1 received a Consistency Score of 100%, and was completely consistent. Subject 2 self-reported a CS GPA of 3.5–3.74, chose answers corresponding to M1 90.48% of the time, and chose answers corresponding to M2 9.52% of the time. Subject 2 received a Consistency Score of 90.48%, showing a slight pattern of deviation. Subject 3 self-reported a CS GPA of 2.5–2.74, chose answers corresponding to M1 63.16% of the time, and chose answers corresponding to M2 36.84% of the time. Subject 3 received a Consistency Score of 63.16%, and exhibited inconsistent behavior. (Due to the nature of the assessment, students did not have to answer each item and may have answered items that did not test the M1 and M2 models. This gives each subject a potentially different denominator.)

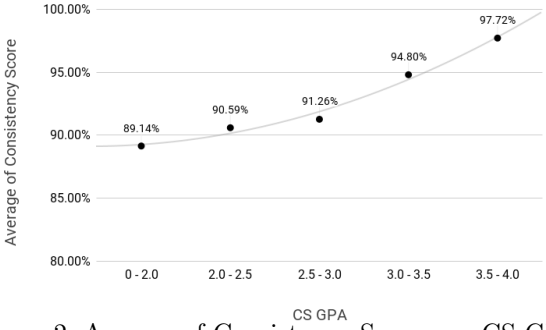


Figure 2: Average of Consistency Scores vs. CS GPA

### 4.2 Consistency Scores vs. Computer Science GPA

Initially, to see if our data showed a correlation between CS GPA and Consistency Scores, we graphed the average Consistency Scores within larger GPA buckets. As seen in Figure 2, the general CS GPA and Consistency Scores show a positive correlation.

To more deeply analyze whether the correlation indicated by the graph is meaningful, we used Spearman's correlation: "Spearman's correlation coefficient, ($\rho$, also signified by $r_s$) measures the strength and direction of association between two ranked variables" [7].

We organized our data set arranging each subject in rank orders of Consistency Score and CS GPA. Owing to the nature of the GPA data and Consistency Scores, we ended up with a number of tied ranks. **These ranks were used to calculate an $r_s$ value of 0.317 as the correlation between the 128 individual Consistency Scores and CS GPA.** This value indicates the existence of a monotonic association [10].

Based on the number of participants (n=128), we calculated the p-value of this correlation to be p=0.00026. Given this result, we can conclude that the null hypothesis, that there is no association between CS GPA and consistency of mental modeling, is extremely unlikely. **Thus we conclude with extremely high likelihood that there is positive correlation between consistency of mental modeling and CS GPA** at our institution.

## 5 Discussion

This section talks about the implications of this study, current practices, and possibilities for moving forward.

### 5.1 Existing Assessments

Assessments currently used on students to determine success only provide insight to the outcome of a student's thinking, rather than the thought process along the way. Because of the nature of introductory computer science courses, the outcome measured on course exams is bound by interpretation and understanding of language-specific syntax. Answers outside of the correct answer are thrown out and not seen as valuable, when in reality, all incorrect answers provide insight. The assessment used in this study aimed to have no correct or incorrect interpretation: all answers revealed the underlying formation of mental models in each subject. The particulars of the syntax are new to all subjects to not advantage those with language familiarity, so all students taking this assessment are on a level playing field.

### 5.2 Consistency and Success

Consistency of mental modeling, or being able to properly follow a mental model, is especially important for computer science students. From our assessment, we can see that consistency is related to academic success within the Computer Science department at our institution. From our experience interacting with CS students in introductory level courses, we have sometimes observed students communicating beliefs that overlap with each other in inconsistent and incongruent ways, preventing them from forming a consistent underlying mental model. This issue spans many different concepts such as variables, assignment, pointers, arrays, recursion, and so on. These inconsistencies inhibit other aspects of learning, such as being able to debug one's own code.

### 5.3 Current Teaching

Most introductory computer science courses are centered around learning and familiarizing language features, rather than generalized concepts. Currently, there is no or limited formal teaching of the notional machine in a typical classroom. Students are expected to form their own mental models and concept of the notional machine through understanding language-specific syntax, and not vice versa. Students that show more than just slight deviation in their consistency results on this assessment, or students that are struggling in their coursework and exams, may need additional support in generalized concepts and mental model building, rather than the typical syntax centered teaching.

### 5.4 Using These Results

The goal of this study was to determine factors to assist intervention programs in improving student success. The instrument we developed could be administered to students early in their computer science degree progress, and used for diagnostic purposes before students have stable or predictive GPAs. Students with low Consistency Scores can still be successful; such a student may benefit from additional support and resources. We want to guide resources to those students who need them the most.

By being able to better understand what factors impact a student's education, universities are able to identify the students with the highest needs before they begin to struggle therefore improving retention rates, raising the average GPA, and having more students successfully graduate.

The assessment used in this study was designed to be easily reproduced and modified to allow other institutions to conduct similar studies on their student populations to determine which factors are prevalent within their own universities as well as across all computer science students as a whole.

## 6 Conclusions and Future Work

### 6.1 Conclusions

We developed a programming assessment that defines a metric of consistency, the Consistency Score. Our assessment employed items that evaluated subjects' consistency of interpretation of operator behavior. Using a Spearman's correlation, we concluded with very high likelihood that there is a positive correlation between students' consistent mental modeling and GPA performance.

This suggests that some lower-performing students may not be using a consistent mental model. Teaching the notional machine and mental modeling may result in higher student achievement and more effective future programmers.

This instrument provides insight into how to better support students and raise academic success.

### 6.2 Future Work

Our goal is to have this study improved based on lessons learned and reproduced both at University of Massachusetts Lowell and at other universities. We suggest the following extensions to strengthen the results of this study: (1) Examine whether mental modeling consistency improves over the course of an undergraduate's career; (2) Revise the programming assessment to more precisely examine additional models; (3) Support other institutions in replicating this study to confirm results and discover trends across institutions.

## References


[1] Jens Bennedsen and Michael E. Caspersen. Failure rates in introductory programming: 12 years later. *ACM Inroads*, 10(2):30–36, April 2019.

[2] Richard Bornat and Saeed Dehnadi. Mental models, consistency and programming aptitude. *Conferences in Research and Practice in Information Technology Series*, 78, 01 2008.

[3] Saeed Dehnadi. Testing programming aptitude. 09 2006.

[4] B. du Boulay. Some difficulties of learning to program. *Journal of Educational Computing Research*, 2:57–73, 1986.

[5] Philip Johnson-Laird. Deductive reasoning. *Annual Review of Psychology*, 50(1):109–135, 1999.

[6] Marcia Linn. The cognitive consequences of programming instruction in classrooms. *Educational Researcher*, 14:14–29, 05 1985.

[7] Lund Research Ltd. Spearman's rank-order correlation, 2018. Retrieved May 2020 from https://statistics.laerd.com/statistical-guides/spearmans-rank-order-correlation-statistical-guide.php.

[8] R. Pea, E. Soloway, and J. Spohrer. The buggy path to the development of programming expertise. *Focus on learning problems in mathematics*, 9:5–17, 1987.

[9] Anthony Robins, Janet Rountree, and Nathan Rountree. Learning and teaching programming: A review and discussion. *Computer Science Education*, 13:137–, 06 2003.

[10] Patrick Schober, Christa Boer, and Lothar A. Schwarte. Correlation coefficients. *Anesthesia & Analgesia*, 126(5):1763–1768, May 2018.

[11] Andreas Stefik and Susanna Siebert. An empirical investigation into programming language syntax. *ACM Transactions on Computing Education (TOCE)*, 13, 11 2013.

[12] Christopher Watson and Frederick W.B. Li. Failure rates in introductory programming revisited. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education*, ITiCSE '14, page 39–44, New York, NY, USA, 2014. Association for Computing Machinery.

[13] Jacqueline Whalley. CSEd research instrument design: the localisation problem. 01 2006.

# The Effects of Mixed Reality Immersion on Users' Performance and Perception of Multitasking While Performing Concurrent Real World Tasks*

*Sarah North[1], Max North [2], David Garofalo[3], and Durgesh Parajapati[1]*
*[1]Computer Science Department*
*[2]Information Systems and Security Department*
*[3]Department of Physics*
*Kennesaw State University*
*Kennesaw, GA USA*
*{snorth, max, dgarofal}@kennesaw.edu*

**Abstract**

The purpose of this preliminary research was to investigate the user's multitasking performance on concurrent tasks while immersed in the mixed reality environment. Twenty-one (n=21) university students, between 18 and 27 years of age, were randomly selected to serve as participants. Each research subject participated in comparative experiments where certain tasks were designed to measure the participant's visuospatial cognition, listening and retention, and logical cognition through arithmetic proficiency while concurrently performing a secondary task. The result of the preliminary experiments indicates the presence of a negative effect on overall effectiveness when multitasking while immersed in a mixed reality environment. Consequently, this research has direct implications for teaching and learning in a variety of educational environments.

# 1 Introduction

Mixed Reality (MR) immersion is defined as a technology that allows users to interface with a digital visual representation of information overlaid on top of the real world (e.g., Microsoft HoloLens is capable of inducing such an environment). However, being immersed in an augmented reality requires a measure of the attention that can distract the user from tasks in the real world, and may possibly be detrimental to the performance effectiveness on both tasks in virtual or real environment. Thus, investigating and testing the user's effectiveness in a specific environment measuring their visuospatial cognition, listening and retention, and arithmetic proficiency.

## 1.1 Technical Terminologies and Definitions

The following terminologies are provided to clarify the basic differences between several comparable, but interrelated technologies associated to this research. Virtual Reality (VR) immerses users in a fully artificial digital environment. Augmented Reality (AR) overlays virtual objects on the real-world environment. Mixed reality (MR) not just overlays but anchors virtual objects to the real world [1, 5, 12, 4].

***Extended MR description*** - MR brings together real world and digital elements. In MR, user interacts with and manipulates both real and virtual objects and environments, using various sensing and imaging technologies. MR allows user to see and immerse herself in the world around her even as she interacts with a virtual environment using her own body (e.g., hands or feet). It provides the ability to have one foot (or hand) in the real world, and the other in an imaginary place, breaking down basic concepts between real and imaginary, offering an experience that can change the way user does tasks [2, 16, 17]. As stated by the Intel [2], "The border between the virtual and real world continues to break down."

## 1.2 Concise Literature Review

One of the perceptible and applied articles reporting distraction levels of mixed reality devices is "Calling while Driving Using Augmented Reality: Blessing or Curse?" which dealt with how distracting augmented reality was when used by the driver of a vehicle for a video call [9]. It revealed that augmented reality provided minimal distraction in the scenario, but offered the caveat that the video feed was placed out of the driver's field of view, so it would require looking away from the road in order to see the person on the other end of the call. The experiment showed that in such a scenario the participants were able to focus on the road and chose not to look at the video feed of the other person, as that

would be dangerous. This article shows that the presence of augmented reality is not inherently distracting, if the application being used can be kept out of the field of view when attention is needed in the real world.

Furthermore, several researchers [8, 7, 14] report measuring visual search and distraction in immersive virtual reality; stating, "Immersive virtual reality provides control over stimuli and conditions at increased ecological validity." The main aim of this research study was to accomplish a transfer of traditional paradigms that assess attention and distraction to immersive virtual reality. Researchers utilized common objects (e.g., objects in a virtual kitchen) as stimuli to increase the ecological validity of exploring attentional effects. The experiment demonstrated evidence of a successful translation of traditional paradigms and manipulations into immersive virtual reality and provided a clear outline for future studies in this field (see [10, 11] for further details).

More specifically, in an investigation with the title of "The Effects of Immersion and Real-World Distractions on Virtual Social Interactions", [13] researchers conducted an experiment. Selected participants interacted with a virtual agent in an immersive virtual environment or non-immersive virtual environment while introduced to three different levels of real-world distractions (i.e., no distraction, passively being exposed to the sound of a ringing cell phone, and actively responding to ringing cell phone). Their verbatim findings suggested that real-world distractions had a negative effect on recognition, recall, and social presence; and increased immersion did not uniformly improve social virtual reality experiences.

A stimulating, and yet practical article with the title of "Multitasking and Prospective Memory: Can Virtual Reality be Useful for Diagnosis?" precisely investigated and demonstrated the utility of virtual reality technology in order to detect prospective memory problems after traumatic brain injury [3]. The results of the experiment showed, as elsewhere [18, 6], that the realization of delayed intention for both experimental and control groups was normal. However, the researchers reported, "TBA the traumatic brain injury participants' executive dysfunctions could be detected in the way they had difficulties in managing well the interference and cognitive overload generated by the multitasking condition."

As the field and applications of the mixed reality expands, the multitasking is becoming one the important components of this environment. In addition, multitasking seems to be growing with advances and availability of tech devices. Multitasking is all around us and frequently contributes to performance of tasks effectively and efficiently. However, there are situations that multitasking may be very harmful. In general, according to Salvucci and Taatgen [15], "... multitasking activities views each activity in terms of the time between task switches, or typical time spent on one task before switching to another."

Hence, in this research study, authors explore the effects of mixed reality immersion on users' experience of multitasking in a virtual environment while performing concurrent tasks in real world. Since this is a preliminary novel research, the research question and null hypothesis have intentionally been kept simple and limited.

***Experimental Research Questions:***

***RQ-1*** *- Does mixed reality immersion have effect on users' performance of multitasking while performing on concurrent real world Tasks?*

***RQ-2*** *- Does mixed reality immersion have effect on users' perception of multitasking while performing on concurrent real world Tasks?*

***Null Hypotheses:***

$\boldsymbol{H_0}$***-1*** *- Being immersed in mixed reality has no significant effect on users' performance of Jigsaw Puzzle multitasking while performing on concurrent real world tasks.*

$\boldsymbol{H_0}$***-2*** *- Being immersed in mixed reality has no significant effect on users' perception of Jigsaw Puzzle multitasking while performing on concurrent real world tasks.*

$\boldsymbol{H_0}$***-3*** *- Being immersed in mixed reality has no significant effect on users' performance of Listening & Retention multitasking while performing on concurrent real world tasks.*

$\boldsymbol{H_0}$***-4*** *- Being immersed in mixed reality has no significant effect on users' perception of Listening & Retention multitasking while performing on concurrent real world tasks.*

$\boldsymbol{H_0}$***-5*** *- Being immersed in mixed reality has no significant effect on users' performance of Mathematical multitasking while performing on concurrent real world tasks.*

$\boldsymbol{H_0}$***-6*** *- Being immersed in mixed reality has no significant effect on users' perception of Mathematical multitasking while performing on concurrent real world tasks.*

## 2 Research Design Methodology

### 2.1 Participants

Twenty-one ($n = 21$) university students both female and male, ages 18 to 27, were selected randomly from different disciplines from two campuses of university to serve as participants. There were disproportionately more males than females (16-5).

### 2.2 Apparatus

The experiments used Microsoft HoloLens to create mixed reality scenarios (see Figure 1). In addition, an array of laptops and other electronic devices were presented to support multitasking and data collections in real-time for the experiment.

Figure 1: A participant wearing HoloLens and immersed in mixed reality while performing real world concurrent multitasking activities.

### 2.3 Instruments for Experimenting Various Multitasks

Three varieties of a *Jigsaw Puzzle* were used for a portion of the visuospatial experiment. These were a 25-piece puzzle designed for younger children, but could provide a challenge to an adult under a short time limit while distracted by a secondary task. The *Listening & Retention* experiments were conducted with a pair of earbuds for use along with the mobile app English Listening Practice. The difficulty of the spoken narration was set to the category Level B1 (Intermediate) of the Common European Framework of Reference for Languages (CEFR). A *Mathematical* assessment was conducted for the arithmetic proficiency portion of the experimenting and measuring with flash cards. The participants had to verbally answer as much addition, subtraction, multiplication, and division questions correctly as they could within the allotted time. Accompanying these three experiments was a secondary task that had to be performed concurrently. Each participant was challenged to maximize the score playing a level in the *Candy Crush Soda* (a popular puzzle game from King, the makers of Candy Crush Saga) twice per experiment on two separate devices.

### 2.4 Procedure for Experiments

Participants were engaged in the experimental phase individually and were briefed regarding the purpose of the study. Participants were asked to fill out

a pre-experiment survey regarding any previous experience with the mixed reality (e.g., Microsoft HoloLens environment), and the game *Candy Crush Soda*. The participants, who did not have an experience with either above environments, were given an ample amount of time to get familiar with both. Whenever participants completed a task before the timer ran out, a bonus point was added to their total score. The bonus points were calculated using the following formula: *Bonus Points = Base Score * Time remaining in seconds /120*.

A control experiment was initiated to collect a baseline data for each task of the study for each participant. Each experiment was limited to two minutes and only the correct answers were counted in their score. The visuospatial cognition experiment challenged each subject to put together a 25-piece puzzle within the time limit. Participants who completed the task before the time limit were given another set of a 25-piece puzzle to solve. For the *Listening and Retention* portion of the control experiment, the participants were required to listen to a 60-second narration twice. The round of *Candy Crush Soda* game would then be stopped for participants who answered five open-ended questions, read to them by the experiment conductor, based on the spoken narration. The participants were asked to verbally answer as many mathematical problems as they could for the arithmetic proficiency control experiment after a flash card with a problem was shown and read to them. The mathematical problems encompassed simple addition, subtraction, multiplication and division. Finally, the participants were asked to play two rounds of *Candy Crush Soda* game within the time limit; on a touchscreen laptop and within a mixed reality while wearing the HoloLens.

The subsequent phase of the study examined the participant's proficiency when asked to complete two unrelated tasks concurrently within a time limit. Participants wore gear (i.e., HoloLens – See Figure 1 in next section) and entered the mixed reality environment while encouraged to take part in the following specific experiments. The first set involved the completion of the visuospatial cognition experimenting with 25-piece jigsaw puzzle while playing a level in *Candy Crush Soda* on a touchscreen laptop and a second time again on a mixed reality environment using the HoloLens. The participants were urged to dedicate equal and alternating attention between tasks. After completion of both experiments, the participants were given a pre-experiment survey with questions about their performance. The next two set of experiments replicated the preceding set of experiments but for the listening and retention assessments and again with the arithmetic proficiency assessment. A set of post-experiment survey questions were administered after the final period was concluded.

## 3 Results and Analysis

### 3.1 Pre-Experiment Survey

The pre-experiment survey data showed the distribution of the prior skills the participants had that could potentially have aided them in this research exploration. Collectively, the participants leaned slightly towards having some proficiency with using the HoloLens. Regardless of their proficiency, all participants were given a chance to get adjusted to the HoloLens' controls. The pre-experiment survey showed that there was an even distribution between participants who were familiar with the game *Candy Crush Soda* and those who were not. Once again, they were given the option to play the tutorial stages of the game before the control experiment could begin. In regard to the distribution of each participant's comfort level with multitasking, the participants were less comfortable with multitasking as a group. Despite the varying levels of expertise the participants had for this study, their results are still viable for the conclusion since their skill level remained the same throughout the experiments.

### 3.2 Candy Crush Soda Analysis

All the participants' scores for the *Candy Crush Soda* game on any particular category, the average scores, the standard deviations (S.D.) and analysis are shown in Table 1 and depicted in Figures 2 and 3. Predictably, there is a significant decrease in performance when comparing the result from multitasking scores with the control experiment (no multitasking) results. The *Jigsaw puzzle* multitasking experiment displayed the most significant drop; a 91% decrease for the HoloLens and an 87% for the laptop category. The second worse performing category is from the *Mathematical* assessment with an 83% drop for the HoloLens and a 50% for the laptop. The emphListening & Retention portion experiment scored better with a 50% drop for the HoloLens and 11% for the laptop.

Out of all the trials that involved the task of playing *Candy Crush Soda*, participants were able to complete the level before the two-minute time limit was up only a few times.

### 3.3 Multitasking Experiment Analysis

The participants filled out a post-puzzle questionnaire before beginning the next experiment set. When asked about the challenges of multitasking with a laptop while solving a *Jigsaw Puzzle*, some of the open-ended responses emphasized an inability to focus on both things at once and the difficulty of switching between the puzzle and the laptop. The same question was asked but with the

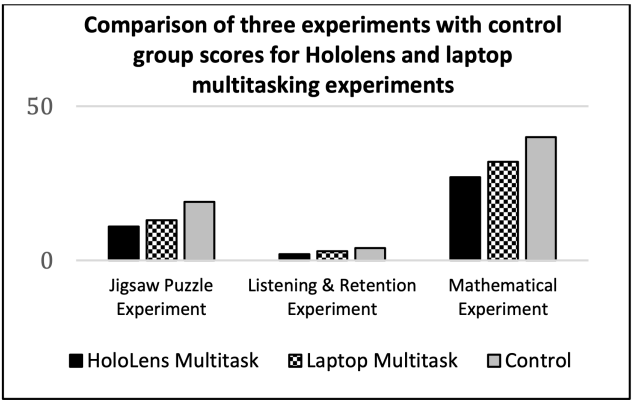



Figure 2: The multiple graph shows comparison of three assessments (*Jigsaw Puzzle, Listening & Retention, and Mathematical*) with control group scores for Hololens and laptop multitasking experiments.

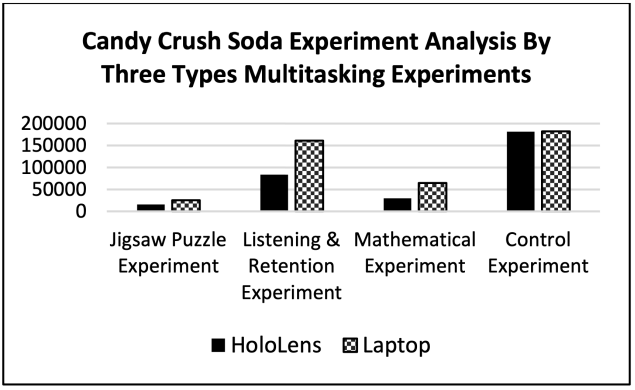



Figure 3: Graph showing comparison of the *Candy Crush Soda* scores between the HoloLens and laptop by three types of multitasking experiments (*Jigsaw Puzzle, Listening & Retention, and Mathematical*). Control experiment column graphs shows non-multitasking experiment.

| Candy Crush Soda Experiment Trials Analysis By Three Types of Multitasking Experiments | | | |
|---|---|---|---|
| **Apparatus/Type** | **Mean** | **S.D.** | **Analysis t-test/p-value** |
| **Jigsaw Puzzle Multitask Experiment Analysis** | | | |
| HoloLens Multitask | 15,809 | 2,993 | *Hololens +Jigsaw vs.* |
| + Jigsaw Puzzle | 11 | 5.93 | *Laptop +Jigsaw* |
| Laptop Multitask | 25,207 | 15,198 | *t*=2.78; *p*=0.0082; *df*=40 * |
| + Jigsaw Puzzle | 13 | 7.83 | *Jigsaw scores with Hololens vs. Laptop* |
| Control [No Multitask] | 95.364 | 77,968 | *t*=0.93; *p*=0.3564; *df*=40 † |
| + Jigsaw Puzzle | 19 | 6.16 | |
| **Listening & Retention Multitask Experiment Analysis** | | | |
| HoloLens Multitask | 83,758 | 7,829 | *Hololens +Listen & Ret vs.* |
| + Listen & Retention | 2 | 1.22 | *Laptop +Listen & Ret* |
| Laptop Multitask | 161,047 | 84,982 | *t*=4.152; *p*=0.0002; *df*=40 ** |
| + Listen & Retention | 3 | 1.10 | *Listening & Retention scores with Hololens vs. Laptop* |
| Control [No Multitask] | 95.364 | 77,968 | *t*=2.789; *p*=0.0080; *df*=40 * |
| + Listen & Retention | 4 | 1.17 | |
| **Mathematical Multitask Experiment Analysis** | | | |
| HoloLens Multitask | 29,898 | 1,486 | *Hololens +Mathematical vs.* |
| + Mathematical | 27 | 8.44 | *Laptop +Mathematical* |
| Laptop Multitask | 64,771 | 24,209 | *t*=6.588; *p*=0.0001; *df*=40 ** |
| + Mathematical | 32 | 7.40 | *Mathematical scores with Hololens vs. Laptop* |
| Control [No Multitask] | 95.364 | 77,968 | *t*=2.041; *p*=0.0479; *df*=40 * |
| + Mathematical | 40 | 6.28 | |

**-Difference statically significant*
***-Difference extremely statically significant*
*†-Difference not statically significant*

Table 1: Depicting the numerical scores analysis (two sampled *t-test; P-value*) of the HoloLens and laptop grouped by the three types of multitasking experiments.

HoloLens experiment. Several participants noted the difficulty of expressing the selection command to the HoloLens headset while another said that the weight of the HoloLens on their head hindered their performance.

For all the participants in the *Jigsaw Puzzle* experiment, 95% of them perceived that the laptop was better suited for *Jigsaw Puzzle* task than HoloLens (5%). Simply, $H_0$-2 was rejected and thus there was a statistically significant difference in perception of users using HoloLens versus laptop.

### Jigsaw Puzzle Experiment Analysis

Reflecting an outcome similar to the *Candy Crush Soda* results, multitasking with the laptop scored higher than multitasking with the HoloLens with only a 36% drop compared to 45% respectively (see Table 1 and Figures 2 & 3).

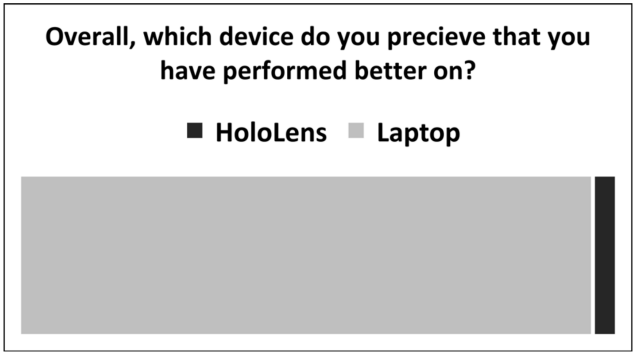



Figure 4: Participants perceived laptop device to be better suited for *Jigsaw Puzzle* experiment over HoloLens.

Further, the statistical analysis of comparison between HoloLens+*Jigsaw* experiment versus Laptop+*Jigsaw* experiment showed a statistically significant difference ($t = 2.78; p = 0.0082; df = 40$), concluding that users performed better with Laptop than HoloLens (immersive mixed-reality). Overall, the hypotheses $H_0$-1 was rejected for mixed reality experiment using *Jigsaw* multitasking, concluding that being immersed in mixed reality has no significant effect on users' performance of multitasking while performing on concurrent real world task (i.e., *Jigsaw Puzzle*). It must be noted that the *Jigsaw* scores were not statistically significantly different ($t = 0.93; p = 0.3564; df = 40$) between mixed-reality and control experiments.

For all the participants in the Jigsaw Puzzle experiment, 95% of them perceived that the laptop was better suited for Jigsaw Puzzle task than HoloLens (5%); (see Figure 4). Simply, $H_0$-2 was rejected and thus there was a statistically significant difference in perception of users using HoloLens versus laptop.

**Listening and Retention Experiment Analysis**

Once again, the results showed that the decrease in the participants' score is less when multitasking on the laptop as opposed to the HoloLens while keeping attention to auditory information. The performance while multitasking with the HoloLens dropped 53% while the drop with the laptop was only 35% (see Table 1 and Figures 2 & 3 in prior section). Moreover, the statistical analysis of multitasking comparison between HoloLens+*Listening and Retention* experiment versus Laptop+*Listening and Retention* experiment indicated that there was a statistically significant difference ($t = 4.152; p = 0.0002; df = 40$), concluding that users performed extremely better with Laptop than HoloLens

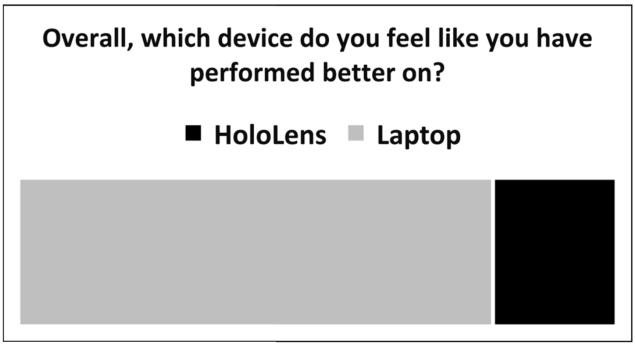



Figure 5: Participants perceived laptop device being better suited for *Listening & Retention* experiment over HoloLens.

(immersive mixed-reality). Overall, the hypotheses $H_0$-3 was rejected for mixed reality experiment using *Listening and Retention* multitasking, concluding that being immersed in mixed reality has significant effect on users' performance of multitasking while performing on concurrent real world task (i.e., listening and Retention). It must be noted that the *Listening and Retention* scores were statistically significantly different ($t = 2.789; p = 0.0080; df = 40$) between mixed-reality and control experiments.

After the *Listening and Retention* experiment was conducted, the participants answered questions on the challenges faced when using the laptop, indicating they had problems focusing and dividing their attention between playing the game and *Listening and Retention*. One participant commented that the questions about the narration itself were too vague. Regarding the multitasking difficulty with the HoloLens, the difficulty focusing and using the HoloLens commands were mentioned again.

While the participants still preferred the laptop as their device of choice for this research, a quarter of the participants perceived that when it comes to *Listening and Retention*, they performed better when using the HoloLens (see Figure 5 above). In this case, $H_0$-4 was rejected and there was a statistically significant difference in perception of users using HoloLens versus laptop.

**Mathematical Experiment Analysis**

Continuing the pattern as that of the previous experiments, multitasking has the least negative effect on tasks performed on the laptop compared to those on the HoloLens when it came to solving simple *Mathematical* problems in experiment. There was a decrease of 35% in performance with the HoloLens and 19% with the laptop (See Table 1 in prior section). Furthermore, the statistical anal-

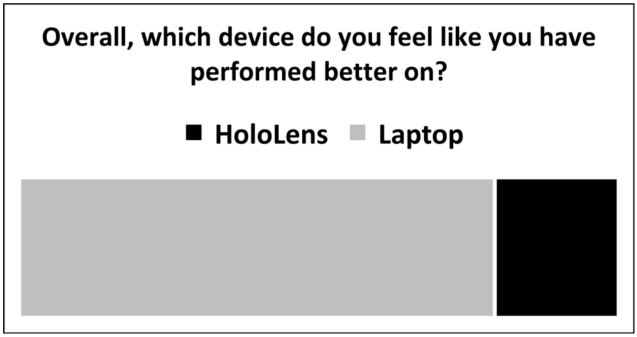



Figure 6: Participants perceived laptop device being better suited for *Mathematical* experiment over HoloLens.

ysis of comparison between HoloLens+*Mathematical* multitasking experiment versus Laptop+*Mathematical* experiment showed a statistically significant difference ($t = 6.588; p = 0.0001; df = 40$), concluding that users performed extremely better with Laptop than HoloLens (immersive mixed-reality). Overall, the hypotheses $H_0$-5 was rejected for mixed reality experiment using *Mathematical* multitasking, affirming that being immersed in mixed reality has significant effect on users' performance of multitasking while performing on concurrent real world task (i.e., *Mathematical*). It must be noted that the *Mathematical* scores were statistically significantly different ($t = 2.041; p = 0.0479; df = 40$) between mixed-reality and control experiments.

Lastly, the participants responded to the post-mathematical survey. Being unable to focus was once again a problem for the participants for both devices. For the laptop, a few participants mentioned that the time spent looking at the math problem was consuming. A similar problem was brought up with the HoloLens when others said that the headset was obstructive when viewing the flashcards.

Similar to the prior result, a majority of the participants still selected the laptop and a quarter chose the HoloLens as their preferred device (see Figure 6). Basically, $H_0$-6 was rejected and there was a statistically significant difference in perception of users using HoloLens versus laptop in this experiment.

### 3.4 Post-Experiment Survey

At the completion of all the experimental portion of the research, the participants filled out the post-experimental survey.

Much of the difficulty that some of the participants faced while using the HoloLens was from the headset not being able to detect their hand gesture

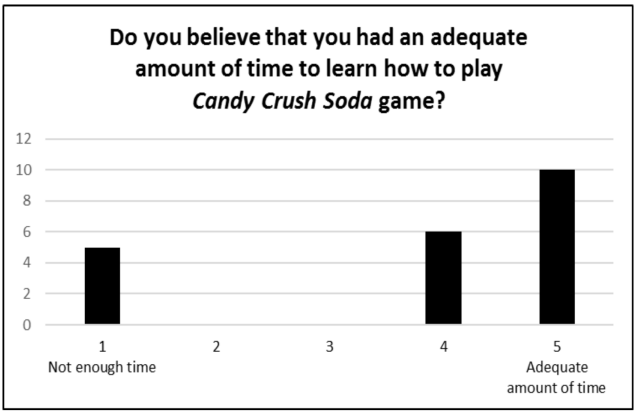



Figure 7: Graph showing the participants responses on whether or not they thought they had received adequate time to learn how to use the HoloLens (scale of 1 to 5).

and therefore failing to register an input command. The one negative question response could be attributed to the participant's unfamiliarity with the headset's quirks. During the experimenting, a few participants realized that the game awarded the player extra bonus points that slightly influenced the scoring giving the participants more points than anticipated. However, the extra bonus points were not statistically significant; thus were not considered in analysis by researchers. The majority of the participants believed that they were given adequate preparation to play *Candy Crush Soda* with 25% not swaying one way or the other based on Figure 7.

When asked about which multitasking experiment gave them the most trouble when accompanied with the HoloLens, the participants were mostly split between the *Jigsaw Puzzle* and the *Listening and Retention* experiment (see Figure 8). This result is supported by the fact that both experiments displayed the most decline in score when compared to each respective control experiment while the *Mathematical* experiment showed the least amount of decrease. Running counter to the result, however, in the accompanying *Candy Crush Soda* multitasking experiment for the *Listening and Retention* showed the least amount of decline compared to the other two multitasking experiments.

As one can surmise, the participants picked the *Mathematical* multitasking experiment as the challenge that gave them the least amount of problems when paired with the HoloLens.

Lastly, the participants were asked an open question on which device they would prefer to use in a professional setting and why. All of the participants

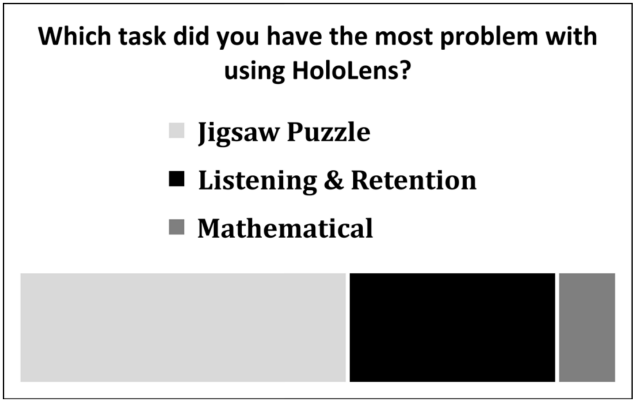



Figure 8: The distribution of multitasks with which the participants struggled most while using the HoloLens.

preferred the laptop as their device of choice. Their response ranged from control and input issues that they' had encountered with the HoloLens to the weight of the headset being too heavy, just to report a few.

As a final note, the complete results and analysis of pre- post-surveys are not reported here. Future research report will be presented by extended and detailed analysis.

## 4 Discussion and Conclusion

This research attempted to answer the general research question of *"Does being immersed in mixed reality have effect on users' performance and perception of multitasking on concurrent real world tasks"*. Analysis of the results from the preliminary rounds of experimenting showed a trend supporting evidence to answer some aspects of the above research question. For all multitasking challenges performed in a mixed reality environment, those results scored consistently lower than multitasking challenges performed in the real world, a common sense expectation. All the participants' performances also supported this assertion as none of them expressed an interest in using the HoloLens in a professional setting. However, due to time constraints, only a limited number of participants ($n = 21$) could complete their experiments and more participants are needed to appropriately support the study's questions and hypothesis.

The time constraints issue was exacerbated by the length of time to complete each trial within each experiment. Possible experimenting fatigue due to the repetitive nature and length of the experiments should also be investigated. Additionally, limitations of an immature technology should also be considered

as the interfacing with the HoloLens headset could potentially be improved with continual hardware and software improvement.

As mentioned in the prior section, during the course of this research, it was discovered that the Candy Crush Soda game could award additional bonus points when the level was solved completely providing a possible uneven result. It is advised that future research should develop two proprietary applications of similar design for the HoloLens and for the laptop that would challenge the participants and yield rational results.

This research investigated three types of multitasking to rigorously experiment the effects of multitasking in a mixed reality environment. For future studies, decomposing this research design into three separate investigations could bring better focus and attention to each area. Consequently, this research has direct implications for teaching and learning in a variety of educational environments.

## Acknowledgment

The initial theoretical and graphical design of visualization component of this research was conducted at Visualization & Simulation Research Cluster/Laboratory that was founded by an equipment grant from the U.S. Department of Defense (DoD)/U.S. Army Research Office (ARO). The content of this work does not reflect the position or policy of the DoD, ARO, NSA, or IDS and no official endorsement should be inferred. Authors thank Andrew Roland, Jonathan Parks, and Rico Santiago for their initial research and assisting in experimental phases of this research. Ultimately, authors express gratitude to all the reviewers for their constrictive and technical suggestions.

## References

[1] The difference between virtual reality, augmented reality and mixed reality. https://www.forbes.com/sites/quora/2018/02/02/the-difference-between-virtual-reality-augmented-reality-and-mixed-reality/.

[2] Virtual reality vs. augmented reality vs. mixed reality demystifying the virtual reality landscape. https://www.intel.com/content/www/us/en/tech-tips-and-tricks/virtual-reality-vs-augmented-reality.html.

[3] Frederic Banville, Pierre Nolin, Sophie Lalonde, Mylene Henry, Marie-Pier Dery, and Rene Villemure. Multitasking and prospective memory: can virtual reality be useful for diagnosis? *Behavioural neurology*, 23(4):209–211, 2010.

[4] Mark Billinghurst and Hirokazu Kato. Collaborative mixed reality. In *Proceedings of the First International Symposium on Mixed Reality*, pages 261–284, 1999.

[5] Răzvan-Alexandru Călin et al. Virtual reality, augmented reality and mixed reality-trends in pedagogy. *Social Sciences and Education Research Review*, 5(1):169–179, 2018.

[6] Tanya Denmark, Jessica Fish, Ashok Jansari, Jignesh Tailor, Keyoumars Ashkan, and Robin Morris. Using virtual reality to investigate multitasking ability in individuals with frontal lobe lesions. *Neuropsychological rehabilitation*, 29(5):767–788, 2019.

[7] Adam Greenfeld, Artur Lugmayr, and Wesley Lamont. Comparative reality: Measuring user experience and emotion in immersive virtual environments. In *2018 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, pages 204–209. IEEE, 2018.

[8] Andrew L Kun, Hidde van der Meulen, and Christian P Janssen. Calling while driving using augmented reality: Blessing or curse? *PRESENCE: Virtual and Augmented Reality*, 27(1):1–14, 2019.

[9] Andrew L Kun, Hidde van der Meulen, and Christian P Janssen. Calling while driving: An initial experiment with hololens.(2017). 2017.

[10] Nilli Lavie. Attention, distraction, and cognitive control under load. *Current directions in psychological science*, 19(3):143–148, 2010.

[11] Nilli Lavie. Attention, distraction, and cognitive control under load. *Current directions in psychological science*, 19(3):143–148, 2010.

[12] Max M North, Sarah M North, et al. Dynamic immersive visualisation environments: enhancing pedagogical techniques. *Australasian Journal of Information Systems*, 23, 2019.

[13] Catherine Oh, Fernanda Herrera, and Jeremy Bailenson. The effects of immersion and real-world distractions on virtual social interactions. *Cyberpsychology, Behavior, and Social Networking*, 22(6):365–372, 2019.

[14] Bettina Olk, Alina Dinu, David J Zielinski, and Regis Kopper. Measuring visual search and distraction in immersive virtual reality. *Royal Society open science*, 5(5):172331, 2018.

[15] Dario D Salvucci and Niels A Taatgen. *The multitasking mind*. Oxford University Press, 2010.

[16] Stephan Schütze and Anna Irwin-Schütze. *New Realities in Audio: A Practical Guide for VR, AR, MR and 360 Video*. CRC Press, 2018.

[17] Christopher Stapleton and Jannick Rolland. Mixing realities at ismar 2009: Scary and wondrous. *IEEE computer graphics and applications*, 30(3):89–95, 2010.

[18] Siobhan Sweeney, Denyse Kersel, Robin G Morris, Tom Manly, and Jonathan J Evans. The sensitivity of a virtual reality task to planning and prospective memory impairments: Group differences and the efficacy of periodic alerts on performance. *Neuropsychological Rehabilitation*, 20(2):239–263, 2010.

# Supporting Computing Accessibility Education Using Experiential Learning Labs*

**Conference Tutorial**

*Saad Khan, Samuel Malachowsky, Daniel Krutz*
*Department of Software Engineering*
*Rochester Institute of Technology*
*Rochester, NY 14623*
*{sk6786,samuse,dxkvse}@rit.edu*

**Abstract**

Our *Accessibility Learning Labs* not only inform participants about how to properly create accessible software, but also demonstrate the need to create accessible software. These experiential browser-based activities enable students, instructors and practitioners to utilize the material using only their browser. This tutorial will benefit a wide-range of participants in the software engineering community, from students to experienced practitioners who want to ensure that they are properly creating inclusive, accessible software. Complete project material is publicly available on the project website: http://all.rit.edu

## 1 Introduction

To fill the existing void in accessibility education, we have created *a comprehensive collection of laboratory activities that are essential to accessibility education.* These labs are collectively referred to as the *Accessibility Learning Labs* (ALL). These systematically developed educational accessibility labs have the primary goals of creating student awareness of the need to create accessible software, and to inform students about fundamental accessibility concepts. The labs are easy to integrate into a variety of existing introductory computing courses (

**example 1.1** *Computer Science I & II) due to their easy to adopt, self-contained nature. No special software is required to use any portion of the labs since*

*they are web-based and able to run on any computer with a reasonably modern browser web browser.*

*The tutorial will provide participants with a mechanism to I) Learn about our provided educational accessibility material, II) Provide feedback and guidance on any necessary modifications to the labs, and III) Share common challenges and best practices for including accessibility in computing education. No prior experience from tutorial participants will be required.*

### *1.1 Lab Structure*

*Each lab addresses at least one accessibility issue and contains: I) Relevant background information on the examined issue, II) An example app containing the accessibility problem, III) A process to emulate this accessibility problem (as closely as possible), IV) Details about how to repair the problem from a technical perspective, and V) Information from actual people about how this encountered accessibility issue has impacted their life.*

## *2 Tutorial Session Agenda*

***Activity 1: Hearing-focused Lab: (30 minutes)*** *This lab instructs participants on proper procedures in making software accessible to users who are Deaf/Hard of Hearing.*

***Activity 2: Colorblindness-focused Lab: (30 minutes)*** *This lab instructs participants on proper procedures in making software accessible to users with colorblindness.*

***Activity 3: Lab Feedback: (15 minutes)*** *Participants will provide feedback on the material, and offer guidance to presenters on the future direction of the labs. This feedback will be incorporated into the design of future labs.*

***Acknowledgements***

*This material is based upon work supported by the NSF under grant #1825023.*

## *References*

*[1] Weishi Shi, Saad Khan, Yasmine El-Glaly, Samuel Malachowsky, Qi Yu, and Daniel E. Krutz. Experiential learning in computing accessibility education. In* Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings*, ICSE '20, page 250–251, New York, NY, USA, 2020. Association for Computing Machinery.*

*[2] Weishi Shi, Samuel Malachowsky, Yasmine El-Glaly, Qi Yu, and Daniel E. Krutz. Presenting and evaluating the impact of experiential learning in*

*computing accessibility education. In* Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering Education and Training, *ICSE-SEET '20, page 49–60, New York, NY, USA, 2020. Association for Computing Machinery. Distinguished Paper Award.*

## *Presenter Biographies*

***Saad Khan*** *is the Project Manager and Developer of the project, dedicated to planning and developing accessibility labs. Saad is a 5th-year Software Engineering Masters student who has led multiple intro CS classes using these labs. He has also led the development of several of the created labs.*

***Samuel Malachowsky*** *is the Co-PI of the project that is developing the described accessibility labs. Malachowsky is a Sr. Lecturer at RIT and has authored 9 pedagogically focused publications. He holds a Project Management (PMP) certification and has authored a textbook on team leadership.*

***Daniel Krutz*** *is the PI of the NSF-funded project [1, 2] that is devoted to creating the presented labs. Krutz has taught approximately ten different graduate and undergraduate software engineering courses and is the author of over twelve pedagogical research papers.*

# Computer Science and Robotics Using Single Board Computers

## Conference Tutorial

*Kevin McCullen and Michael Walters*
*Computer Science and Physics Departments*
*State University of New York College at Plattsburgh*
*Plattsburgh, NY 12901*
*kmccu006@plattsburgh.edu*

Single Board Computers (SBCs) are a key enabler of the Maker movement and have found a place in education. The two dominant SBCs in education, the Arduino and Raspberry Pi, are now commonly used in courses in Engineering, Computer Science, and Robotics [4, 3, 2, 5]. Their ability to inexpensively work with sensors, servos, Wifi, Bluetooth, audio, video, and images opens opportunities for exciting and engaging projects [1]. New, more powerful and/or cheaper platforms continue to be added to the SBC landscape. Examples include the micro:bit, Arduino MKR series, Adafruit Feather, and Teensy.

SBCs are not without challenges for instructors. Most are consumer or hobbyist grade hardware, product development cycles can be very short leading to limited lifespans for specific models, documentation can be limited, and the number of options available can make it difficult to select a path.

These systems also present challenges for students. There are extensive pools of information available on the internet concerning these devices. However, it is not always correct or easily understood. Students need to learn to synthesize various information streams to create a viable solution.

Our tutorial will cover practical aspects of using these systems for education. This tutorial is targeted at faculty who are interested in using SBCs, or who have made their first foray into their use. Some of the key topics to be discussed include the diversity of SBCs and their relative pros and cons; with specific attention to price points, configuration and deployment, sensor and servo compatibility, and the differing programming models used by different SBCs (for example, full Linux install versus limited resource microcontrollers.) We will also cover aspects of working with pulse width modulation controlled servos in both environments, remote control of these systems, audio processing, and video/image processing using OpenCV.

Because of the different strengths of the systems, it is often necessary for projects to use several SBCs. We will discuss the options for combining SBCs, techniques for communication between SBCs (for example in a robot system) and potential pitfalls. The tutorial will also include demonstrations of projects using a variety of SBCs, with code examples in Circuit Python, Python, and C.

This is an update and revision of a tutorial given three years ago, updated to reflect more learning and newer systems that have since become available.

## Biography

Dr. Michael Walters is an Associate Professor of and Chair of Physics at SUNY Plattsburgh. He is in charge of the hardware component of the new Robotics major at SUNY Plattsburgh. Dr. Walters designed KIF (Keep It Fun); a robotics learning platform; based on 3-D printed components, Arduino, and Raspberry Pi. KIF is in its third generation, and is used in introductory and intermediate robotics laboratories. Dr. Walters mentored a team of students who competed in the NASA Centennial Challenge: Robotic Sample Return Challenge, using Raspberry Pis and Arduinos as the central controller, vision processing, and motion controllers for their rover. He currently is researching connecting a robot controlled by an Arduino to an Amazon Dot to allow for voice control of an electric wheelchair. Dr. Walters also works with youth in informal science education, using 3-D printers and robotics.

Dr. Kevin McCullen is an Associate Professor of Computer Science at SUNY Plattsburgh. Dr. McCullen has designed and taught a course in Embedded Systems using the Raspberry Pi and the Sense HAT. Additionally, Dr. McCullen is the recipient of a university grant to use Arduinos and Raspberry Pi systems for teaching and undergraduate research. Dr. McCullen's projects include using an Arduino with sensors for classes in networking, undergraduate research in computer vision using OpenCV, and taking attendance using Wifi and Bluetooth. Dr. McCullen participates extensively in informal science education, working with Raspberry Pis and as a judge and advisor for First Tech Challenge robotics.

Drs Walters and McCullen teach courses in the SUNY Plattsburgh Undergraduate Robotics major, which has graduated its first class of students. The major is built on using SBCs and microcontrollers.

## References


[1] Leah Buechley and Michael Eisenberg. The lilypad arduino: Toward wearable engineering for everyone. *IEEE Pervasive Computing*, 7(2):12–15, 2008.

[2] Jake Rowan Byrne, Katriona O'Sullivan, and Kevin Sullivan. An iot and wearable technology hackathon for promoting careers in computer science. *IEEE Transactions on Education*, 60(1):50–58, 2016.

[3] Kevin McCullen. Teaching embedded systems using the raspberry pi and sense hat. *Journal of Computing Sciences in Colleges*, 33(3):148–156, 2018.

[4] Septimiu Mischie. On teaching raspberry pi for undergraduate university programmes. In *2016 12th IEEE International Symposium on Electronics and Telecommunications (ISETC)*, pages 149–153. IEEE, 2016.

[5] M Cristina Rodriguez-Sánchez, Angel Torrado-Carvajal, Joaquin Vaquero, Susana Borromeo, and Juan A Hernandez-Tamames. An embedded systems course for engineering students using open-source platforms in wireless scenarios. *IEEE transactions on education*, 59(4):248–254, 2016.

# Short Modules for Introducing Heterogeneous Computing*

**Conference Tutorial**

*David P. Bunde[1], Apan Qasem[2], Philip Schielke[3]*
*[1]Knox College*
*Galesburg, IL 61401*
dbunde@knox.edu
*[2]Department of Computer Science*
*Texas State University*
*San Marcos, TX 78666*
apan@txstate.edu
*[3]Concordia University Texas*
*Austin, TX 78726*
Philip.Schielke@concordia.edu

## Abstract

CS faculty have spent the last several years adding parallel computing to their curricula since essentially all processors sold today have multiple cores. A typical target system is a multicore processor with identical cores. This is the configuration for most current desktop and laptop systems, but the technology continues to evolve and systems are incorporating heterogeneity. Many phone processors include cores of different sizes so the phone can vary its power and performance profile over time. Other processors incorporate low-power modes or instructions for specialized computations. Meanwhile, high-end systems make heavy use of accelerators such as graphics cards. We are at a stage where heterogeneous computing concepts should pervade the curriculum rather than being limited to upper-level courses.

This tutorial motivates heterogeneous parallel programming and then presents modules that introduce aspects of it such as ener-

gy/performance tradeoffs, SIMD programming, the benefit of memory locality, processor instruction set design tradeoffs, and CPU task mapping. Each module uses only a few days of class time and includes assignments and/or lab exercises which are available online (https://github.com/TeachingUndergradsCHC/modules/). Here are the modules:

1. The first module shows the challenges and benefits of task mapping on a heterogeneous system. The module includes a lab to provide students with hands-on experience running parallel workloads in heterogeneous environments. It is aimed at CS 2, but also fits in Systems and Parallel Programming courses.

2. The second module looks at heterogeneity on ARM processors, particularly Thumb mode, a low-power mode with restricted instructions. The module is based on the Raspberry Pi, a low-cost system aimed at hobbyists. It highlights performance/power tradeoffs and is aimed at Computer Organization.

3. The third module shows how memory locality can improve performance on a program that uses CUDA to run on a graphics processing unit (GPU). This module demonstrates heterogeneity resulting from both CUDA's SIMD model of computing and the different memory types on a GPU. It highlights memory locality and is aimed at systems-oriented courses.

## Acknowledgements

This tutorial presents work supported by NSF grants OAC-1829644 & OAC-1829554.

# Building and Hacking an Exploitable WiFi Environment for Your Classroom – Even for Remote Participants*

**Conference Workshop**

*Ahmed Ibrahim*
*University of Pittsburgh*
*Pittsburgh, PA*
*aibrahim@pitt.edu*

With the widespread of WiFi nowadays, it is important to show students how WiFi access points can be exploited in practice. The theory behind exploiting WEP and WPA2 has been available for a number of years. However, it has not been easy to offer students the opportunity to apply these theories in a real environment. In this workshop, you will learn how to build and configure WiFi access points for your students to hack. You will learn how Raspberry Pis can be used to act as the necessary clients for those access points and you will have access to the Raspberry Pi scripts and all access point configuration directions. We will go over setting up a VPN with a Raspberry Pi for each student to use for attacks. Then, we will discuss the different hacking scenarios where WEP access points have connected clients versus no connected clients. In addition, you will get a chance to hack up to four WEP access points in addition to a WPA2 access point. *Note: Participants must have a computer with at least 25GB of disk space available and VirtualBox installed.*

The presenter has a physical environment dedicated to the workshop which includes WiFi access points. Participants will connect to the presenter's physical environment via a tested and reliable VPN (as shown in figure 1). Once a participant connects to the VPN they will be able to SSH into a Kali Linux dedicated machine (one machine per participant) which is equipped with all necessary packages and hardware to engage in the workshop activities within the sandboxed WiFi environment built for this workshop.

A few days before the workshop, the presenter will email the participants with links to (a) download the Virtual Machine required for the workshop and

(b) setup their VPN connection such that they can engage in the workshop remotely. The instructor has offered the described activity in an undergraduate security course during the Spring 2019, Fall 2019, Spring 2020, and Fall 2020 semesters and received positive feedback about its educational benefit.

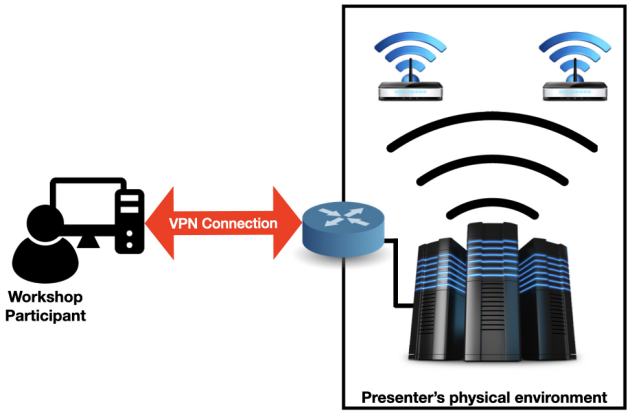



Figure 1: WiFi Environment behind the Presenter's VPN

# COVID-19 Data Analysis Applied to Computer Science Courses*

**Faculty Poster**

*Kehan Gao and Sarah Tasneem*
*Computer Science*
*Eastern Connecticut State University*
*Willimantic, Ct 06226*
*{gao,tasneems}@easternct.edu*

At Eastern Connecticut State University, we include or teach data analysis in different levels of computer science courses, including our Computer Science (CS) gateway course CSC180: Fundamentals of Computing, two programming courses; CSC202: Introduction to Programming and Machine Intelligence and CSC203: Advanced Programming for Data Science, as well as CSC305: Data Mining and Applications. In all these courses, students will learn data science technologies and develop data analysis skills appropriate to the course level. For example, CSC180 provides students with general concepts of data science and the role of computer science in data analysis. CSC202 and CSC203 put more emphasis on programming techniques such as classes, objects, methods, file I/O and other packages involved. Students will develop skills to apply them to data processing and analysis. CSC305 explores the entire data mining (or analysis) process and focuses on stages ranging from data collection and preparation to data processing and modeling until pattern and knowledge discovery.

As the whole world has been suffering from a pandemic, COVID-19 data analysis is a fitting project that can be used at different levels of CS courses, which involve data science. Comparing with other data, COVID-19 data possesses the following advantages: 1) Data availability: COVID-19 data can be obtained from many well-known public sources such as CDC [1], JHU [2], etc. 2) Ease of comparison: Many analysis results and data visualization outcomes have been published based on these common COVID-19 data, so students can compare their own results with the published results. 3) Practicality: Since COVID-19 data is real-world data, it provides students with the opportunity to use the technology and skills learned in the classroom to deal with such dynamic and constantly updated datasets and gain benefits. 4) Multi-perception:

this data is multi-dimensional, including information about counties, states, and countries. Also, it contains information about age, gender and race. Multiple facets of data allow data analysis to be performed from different angles.

In CSC180, when assigning the COVID-19 data analysis project, we provided daily new case data of CT, MA, NJ, NY and US in the csv table format. Students were required to use certain tools (e.g., Python) to show the trend of new cases in each state and the country and compare differences. To prepare them for the project, students were provided with simple data sets and program examples. Guidelines were given on how to read formatted data and display it in a specific graphical representation. In CSC202, we provided students with unemployment data [3] (e.g., the insured unemployment rate) of Northeast states (CT, MA, ME, NH, NJ, NY, PA, RI and VT) and asked them to discover changes in unemployment during the pandemic last year. They used tables and graphics to present results and saved them in files and appropriate directories. In CSC203, a possible data analysis project could be to create a simple COVID-19 dashboard. The procedure is as follows: 1) Collect data from trustworthy sources; 2) Clean, select and prepare data according to the dashboard requirements; 3) Perform data processing, such as using Plotly to visualize 10 worst-hit states; or using Folium to plot all confirmed cases (or deaths) on the US map; 4) Use Voilà to convert the Notebook to a standalone dashboard. In CSC305, possible project topics include "How does the COVID-19 vaccine affect the number of positive cases?" and "Assess the potential relationship between the number of administered COVID-19 vaccines and COVID-19 mortality". Students need to collect and prepare data based on a specified problem outlines and process the data accordingly. They may spot certain patterns and interesting trends and draw informative conclusions.

This work aims to share the experience of applying COVID-19 data analysis to different levels of computer science courses that teach data science. Interested parties can gain some useful insights from the report.

## References


[1] CDC COVID Data Tracker. https://covid.cdc.gov/covid-data-tracker/\#cases_casesinlast7days.

[2] COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University. https://github.com/CSSEGISandData/COVID-19, 2020.

[3] Employment and Training Administration. https://oui.doleta.gov/unemploy/claims.asp, May 2020.

# Cybersecurity Virtual Summer Workshop for Secondary School Teachers: An Experience Report*

**Faculty Poster**

*Sarbani Banerjee and Neal Mazur*
*Department of Computer Information Systems*
*State University of New York at Buffalo State*
*Buffalo, NY 14222*
*{banerjs,mazurnm}@buffalostate.edu*

Throughout the past decade the workforce demands are rising steadily in the field of cybersecurity, outstripping the supply of skilled workers [6]. Cybersecurity education is the backbone of building strong cybersecurity professionals and informed citizens and the need for bringing cybersecurity education for the K-12 students is rapidly increasing [3, 4]. With the long-term goal of broadening participation in field of cybersecurity, the presenters offered a four-day virtual summer workshop for Western New York (WNY) teachers at Buffalo State College in the summer of 2020 [2]. More than 25 middle and high school teachers primarily from the WNY area attended the workshop.

The workshop curriculum included various topics of cybersecurity such as 'Cyber Ethics', 'Ethical Hacking', 'Cyberbullying', 'Network Security', 'Internet of Things', 'Web Security', 'Social Engineering', 'Data Security and SQL Injection', 'Password Protection and Online Safety', 'Cryptography', 'Raspberry PI', 'Python Programming' as well as cybersecurity concepts and first principles [1]. In this poster, we share our curriculum, present data on short-term impacts, and describe our in-progress work to evaluate the workshop's longer-term impacts.

Since the participants in the workshop were all middle and high school teachers, the curriculum had a focus on material appropriate and important to their students. A unit on ethical behavior was essential in this regard. Teachers were able to explore and discuss topics of cyberbullying and cyberstalking, plagiarism and copyright infringement, and the relationship between moral behavior online and offline. Technical material such as cryptography was covered

at different levels of sophistication emphasizing examples and unplugged activities. Raspberry Pi (programmed in Python) was used as platform for demonstrating cybersecurity concepts which lends itself to many hands-on robotic and other electronic activities to interest younger students.

Since the cybersecurity skills shortage is reaching widespread proportions, one way to ensure a larger pipeline in cybersecurity is to train more middle and high school teachers to not only teach cybersecurity in their schools or integrate cybersecurity concepts in their classrooms but also to promote cybersecurity as an attractive career path [5]. To this effect the summer workshop offered a lengthy 'Cybersecurity Careers and Awareness' session with guest speakers from the industry sharing their experiences in working as cybersecurity professionals. On the last day of the workshop series the teacher participants shared Lesson Plans they created based on the cybersecurity topics presented at the workshop. Participants worked in teams of two preparing lessons during the first three days of the workshop through Zoom breakout rooms. A major drawback associated with virtual conferences is the lack of opportunity to share experiences and network with other participants. The use of these two person teams gave the teachers an opportunity to interact more personally with a colleague. The presentations themselves were a gratifying experience for the participants since they not only developed cybersecurity lesson plans for their students but learned from lesson plans and presentations of the other teachers. Ample time was given following each presentation for discussion, constructive comments, and sharing of experience. There are some benefits associated with virtual delivery of the workshop. Teachers not only from WNY, but from all over New York State attended the workshop, including one international teacher from Turkey. Teachers being able to attend from home gives them more flexibility concerning their schedule associated with travel and the ability to revisit the recorded workshop.

A thirty-question survey instrument was developed and administered to all the participants on the final day of the workshop. This survey questionnaire was designed to assess various components of the workshop including the effectiveness of the workshop in preparing the participants to teach/coach cybersecurity in their schools. The findings will be presented in detail in the poster.

## References


[1] Cybersecurity concepts and first principles of cybersecurity. https://cs4hs.buffalostate.edu/sites/cs4hs.buffalostate.edu/files/uploads/Documents/Summer%20Workshop%20Agenda/Cybersecurity%20First%20Principles.pdf.

[2] Cybersecurity summer workshop. https://cs4hs.buffalostate.edu/cybersecurity-summer-workshop.

[3] National initiative for cybersecurity education (nice) cybersecurity workforce framework. https://www.cisa.gov/nice-cybersecurity-workforce-framework.

[4] The state of cybersecurity education in k-12 schools. https://cyber.org/news/state-cybersecurity-education-k-12-schools.

[5] Cybersecurity career paths and progression. https://niccs.cisa.gov/sites/default/files/documents/pdf/cybersecurity%20career%20paths%20and%20progressionv2.pdf?trackDocs=cybersecurity%20career%20paths%20and%20progressionv2.pdf, 2019.

[6] Strategies for building and growing strong cybersecurity teams: Cybersecurity workforce study. https://www.isc2.org/-/media/ISC2/Research/2019-Cybersecurity-Workforce-Study/ISC2-Cybersecurity-Workforce-Study-2019, 2019.

# Exploring Direct Simulation Monte-Carlo Techniques for Science Applications*

**Faculty Poster**

*Vladimir V. Riabov*
*Department of Mathematics and Computer Science*
*Rivier University*
*Nashua, NH 03060*
vriabov@rivier.edu

The revision of STEM education curricula provides computer-science faculty with the opportunities of introducing the advanced computing methods in the courses related to applications in various challenging areas of sciences and technologies. Even basic background of students in mathematical statistics, theory of probability, and molecular physics could help them in simulating real physical phenomena in gases and plasmas using Direct Simulation Monte-Carlo (DSMC) methods. The introduction of these methods to students and results of their exploration projects focusing on various practical applications are reviewed in this paper.

The development of the DSMC techniques and their applications for predicting aerothermodynamics characteristics of space vehicles during flights in the upper atmospheric layers of planets were widely discussed at numerous symposia and conferences for the last 60 years. The DSMC methodology and its implementation in studies of rarefied-gas flows were described in [1, 2, 3]. Samples of computer programs (written in Java, C, FORTRAN, and other languages) can be found in [1, 2, 3, 4]. These codes were modified by students and used in their projects for solving various problems in molecular physics, thermodynamics, and aeronautics.

The DSMC techniques are covered in several CS courses including Discrete Mathematics, Numerical Methods, System Simulation & Modeling, Software Engineering, and the Advanced CS Projects. The traditional topics of statistical analysis and probabilities are covered during the first three classes with the focus on the importance of the sample sizes and asymptotic analyses. The validity of Monte-Carlo simulations is studied for the "classical" cases of numerical

estimations of twofold integrals, values of the error function, and surfaces and volumes of simple-shape bodies. The acquired knowledge and skills are used by students in mini-case studies of molecular collisions, estimations of transport coefficients in gases [2, 8], structures of shock waves, and aerodynamics of simple-shape probes [7]. All these cases cannot be solved by applying analytical or numerical approaches in solving ordinary and/or partial differential equations that are simply not applicable for studying these molecular phenomena.

In final capstone projects, students investigate relaxation processes in expanding gas jets [9], separation of gas-mixture components in flows near blunt bodies, the reverse Magnus effect on a rotating cylinder, aerodynamics of toroidal-shape balloon parachute [5], shock-shock interactions, interference between bodies in hypersonic flows [10], the bulk viscosity and various energy relaxation parameters. This last group of studies requires the creative development and adaptation of DSMC methods by students in estimating multivariable and multi-fold integrals that cannot be solved analytically. The results of these studies have been presented at conferences and summarized in the author's articles [8, 7, 6].

In the course evaluations, students stated that they became deeply engaged in case studies and project activities through examining the challenging problems related to the real-world applications of the state-of-the-arts computing technologies in aeronautics and applied physics.

## References


[1] Graeme Austin Bird. Molecular gas dynamics. *NASA STI/Recon Technical Report A*, 76:40225, 1976.

[2] Graeme Austin Bird. Molecular gas dynamics and the direct simulation of gas flows. *Molecular gas dynamics and the direct simulation of gas flows*, 1994.

[3] Graeme Austin Bird. *The DMSC Method*. CreateSpace Independent Publishing Platform, 2013.

[4] AV Kashkovskyt. Computational tools for rarefied aerodynamics. *Rarefied gas dynamics: Space science and engineering*, page 115, 1992.

[5] James Moss. Dsmc simulations of ballute aerothermodynamics under hypersonic rarefied conditions. In *38th AIAA Thermophysics Conference*, page 4949, 2005.

[6] Vladimir V Riabov. Numerical and experimental simulation techniques in hypersonic low-density aerothermodynamics. In *23rd AIAA International Space Planes and Hypersonic Systems and Technologies Conference*.

[7] Vladimir V Riabov. Comparative similarity analysis of hypersonic rarefied gas flows near simple-shape bodies. *Journal of Spacecraft and Rockets*, 35(4):424–433, 1998.

[8] Vladimir V Riabov. Gas dynamic equations, transport coefficients, and effects in nonequilibrium diatomic gas flows. *Journal of Thermophysics and Heat Transfer*, 14(3):404–411, 2000.

[9] Vladimir V Riabov. Kinetic phenomena in spherical expanding flows of binary gas mixtures. *Journal of thermophysics and heat transfer*, 17(4):526–533, 2003.

[10] Vladimir V Riabov. Numerical study of interference between simple-shape bodies in hypersonic flows. *Computers & structures*, 87(11-12):651–663, 2009.

# Student-made Online Discrete Math Drills*

**Lightning Talk**

*Sebastiaan J. C. Joosten*
*Department of Computer Science*
*Dartmouth College, Hanover, NH 03755*
*Sebastiaan.Joosten@dartmouth.edu*

**Introduction.** This paper presents a set of online Discrete Mathematics exercises, and describe how they were made. The motivation is threefold: First, to spread the word about an online environment for doing Discrete Mathematics exercises, in the hope that others might find it useful for their courses. Second, to describe experiential learning in a Functional Programming class. Third, to invite feedback on Discrete Mathematics exercises and suggestions for additional exercises.

**Motivation for online Discrete Mathematics exercises.** A key way in which students learn concepts of Discrete Mathematics is by completing practice exercises. In-person, students can be asked if they practiced, but with the increased prevalence of remote learning, an online environment was created to support the following setup: Online practice exercises are graded homework, students can complete exercises with immediate feedback until they have completed ten in a row correctly. Either full or no points are awarded and no manual grading is involved. The environment is integrated with Canvas.

Discrete Mathematics consists of different topics: set-theory, combinatorics, probability, graphs and numbers. We want basic exercises for all topics. We asked students taking the Functional Programming course to create them.

**Experience with the Functional Programming course.** Students learned Haskell during the first half of the course following the textbook by Richard Bird [1], as during an earlier iteration of this course. For the second half of the course, students were divided into seven pairs, and each got to pick one

out of twelve 'simple' assignments. These assignments helped students understand the framework within which the exercises were shown. Next, students got to pick from 'proof-based' assignments in fresh groups. These assignments had students generate a derivation and create a Discrete Mathematics exercise based on it. For instance, the Discrete Mathematics student can be asked to put the steps in the right order. Functional Programming students added the exercises through Git pull-requests as a way of submitting their work. On the final day of the course, groups presented their work and they could try each other's exercises on the departmental server.

The Functional Programming students were enthusiastic about programming something that would actually be used. At the time of writing, student evaluations for the Functional Programming course have not been completed yet, and the Discrete Mathematics course has not taken place yet (this will take place in December and from January to March respectively).

**On the online exercise environment.** The environment consists of generic JavaScript and HTML code to render exercises and feedback, and send responses back to the server. Server-side, the environment is integrated with Canvas. All code is provided to Functional Programming students together with a way to run a single-user exercise server locally. The code is available online at: https://github.com/sjcjoosten/cs30
The application can be tried at: https://cs.dartmouth.edu/~sjc/cs30/

## Biography

Sebastiaan Joosten is a Lecturer at Dartmouth. Sebastiaan has a Masters in Discrete Mathematics from Twente University, a PhD in Computer Science from Eindhoven University of Technology (both in the Netherlands). He was previously a PostDoc at the University of Innsbruck (Austria) and an Assistant Professor at Twente University. His research focuses on Automating Correctness, operating at the boundaries of Logic, Mathematics, and Computer Science, and he likes bringing his research into the classroom.

## References

[1] Richard Bird. *Thinking functionally with Haskell.* Cambridge University Press, 2014.

# Pedagogical Best Practices for Teaching Foundational Computer Science Courses in Alignment with Employer Technical Interviews*

**Panel Discussion**

*Robert J. Domanski*
*Director of Higher Education*
*NYC Tech Talent Pipeline*
*City of New York*
*rdomanski@sbs.nyc.gov*

## 1 Abstract

In 2017, New York City Mayor Bill de Blasio's Tech Talent Pipeline (TTP) industry partnership launched the "CUNY 2x Tech" initiative – a $20 million investment into Computer Science programs within the City University of New York (CUNY). The goals of the initiative were to grow the City's tech workforce by doubling the number of tech Bachelors degrees awarded annually by 2022, increase the employability of such graduates to position them for success in connecting to full-time jobs in the field at market-rate salaries, and increase the rate at which the NYC tech industry consistently draws from this pool of talent in their hiring practices.

While a multi-pronged strategy for achieving these goals has been implemented – which includes the provision of additional faculty lines, industry adjuncts, academic advisement, career coaching, and a robust internship program – industry feedback was gathered highlighting that even strong students often struggled to pass employer hiring screenings because their foundational CS knowledge, and the ways in which such knowledge is often assessed in the classroom, was often misaligned with the types of questions employers ask on those foundational topics in technical interviews.

To address this challenge, a new model for a Faculty Workshop was developed to equip CUNY CS faculty with industry insights, pedagogical Best Practices, and community support so that they could integrate direct career preparation for students into their teaching methods and instructional design. The workshop prioritized true interaction among the faculty, with insights and Best Practices being shared and ideated upon by all participants with their peers.

Our proposal is to host a panel discussion at CCSCNE 2021 representing different perspectives of various stakeholders who participated in this "CUNY 2x Tech" Faculty Workshop. After the panel discussion and subsequent Q&A, we expect the audience of faculty will have renewed ideas for how foundational CS topics can be pedagogically framed for students in ways that better align with – and set students up for success in – employers' technical interviews.

### 1.1 Panelists

The composition of the panel will include two full-time CUNY CS faculty members who participated in the workshop, a representative from industry who can speak in detail about the characteristics of employers' technical interviews and perceived student "gaps", and a representative from the City government who can speak to the workshop's design considerations.

### 1.2 Structure

We propose a 60-minute program with four speakers given ten to fifteen minutes each to speak. The Moderator will summarize panel talks for the audience and have two to three planned questions to start the audience discussion. Fifteen to twenty minutes will be provided for questions and answers.

### 1.3 Postition Statement - Panel

Repeated analyses performed by TTP and its "CUNY 2x Tech" initiative have highlighted that CUNY CS graduates tend to be fairly strong in foundational topics, however that strength often fails to translate well in employer-based assessments on those very same topics. From a pedagogical perspective, there are many - often subtle - changes that faculty can incorporate into their teaching methods and instructional design to deliver their foundational courses in ways that not only effectively teach the course content but also align with how employers assess those same knowledge areas. Often, these are rather subtle, light-lift approaches such as re-thinking how exam questions are designed and assessed, or in requiring Code Reviews or whiteboarding activities as part of

homework or lab assignments; other times, these can be more significant reforms such as re-weighting the value of exams vs. project builds in calculating final course grades. The "CUNY 2x Tech" Faculty Workshop sought to surface such pedagogical Best Practices in conjunction with industry input, and our position is that the resulting insights - to be discussed in this panel - will aid in positioning students for success in landing full-time employment in the field after graduation.

## 2 Panelists

**Eva Sofianos** is a Lecturer of Computer Science and Deputy Chair at Lehman College (CUNY). Eva is passionate about technology, loves teaching and believes in community giving back. Prior to joining Lehman she spent over a decade developing software solutions at IBM. She worked briefly for Weill Medical Center of Cornell University, as well as some NYC startups, and taught at the American Museum of Natural History in NY, leading an all girls BridgeUp STEM Cohort. Some career highlights prior to joining Lehman include co-authoring an IBM Redbook for Private Clouds, co-founding a technical training program named Traincube, and co-founding a community known as GDG Bronx. Since joining Lehman as a full-time faculty member in 2018, she has acquired multiple grants for curriculum redesign and project-based learning.

**Steven Fulakeza** is a Lecturer of Computer Science at Lehman College (CUNY), where he has been a full-time faculty member since 2018. He received his M.S. degree in Computer Science in 2017 and B.S. degree in Computer Science in 2013, both from Lehman College. Steven has also served as an instructor for the NYC Tech Talent Pipeline (TTP). His teaching interests include operating systems, database systems, introduction to networks, programming, and web development. He received a CUNY Tech Innovations Award in 2016 and City College of New York Behind the Scenes Information Technology, Excellence, Achievement and Merit Award in 2017.

**Nikolai Avteniev** is a Senior Staff Software Engineer at LinkedIn. After graduating with a Bachelor of Science in Computer Science from Brooklyn College (CUNY), Nikolai started his professional career in software at JPMorgan Chase. After earning a Master of Science in Computer Science degree from New York University (NYU), Nikolai took the experience of building and running an Agile development team to Real Time Risk Systems where he was one of the two founding engineers. While there Nikolai leveraged the agile software engineering values and practices to build a successful software product that helped hedge funds manage multi-billion-dollar portfolios in near real-time.

Next, Nikolai joined a New York City AdTech start-up Intent Media, where he experienced Agile software development applied in a larger organization with multiple teams coordinating to deliver new product features at a furious pace. Currently Nikolai is working at LinkedIn in the NYC (New York City) engineering office. Nikolai also has developed and taught a Modern Software Engineering course numerous times as an industry adjunct at the City College of New York (CUNY) as part of the NYC Tech-in-Residence Corps program.

**Robert Domanski**, Ph.D., is the Director of Higher Education for the New York City government's Tech Talent Pipeline industry partnership. Rob oversees the "CUNY 2x Tech" initiative, a $20 million investment by the City to double the university's number of tech-related Bachelor's degrees awarded by 2022. Rob also previously spent 13 years teaching Computer Science at the College of Staten Island (CUNY), has certificates in pedagogy and instructional design, and currently teaches Python and Web Development as an adjunct at Kean University purely because he loves teaching and working with students.