The Journal of Computing Sciences in Colleges

Papers of the 26th Annual CCSC Northeastern Conference

April 1st-2nd, 2022
Pace University
Pleasantville, NY

Baochuan Lu, Editor
Southwest Baptist University

Jeremiah W. Johnson, Regional Editor
University of New Hampshire

Volume 37, Number 8 **April 2022**

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.

Copyright ©2022 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

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 (2024), (417)328-1676, blu@sbuniv.edu, Southwest Baptist University - Division of Computing & Mathematics, 1600 University Ave., Bolivar, MO 65613.

Brian Hare, Treasurer (2023), (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(2024), 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 (2024), (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 (2024), (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.
George Dimitoglou, Comptroller, (301)696-3980, dimitoglou@hood.edu, Dept. of Computer Science, Hood college, 401 Rosemont Ave. Frederick, MD 21701.
Carol Spradling, National Partners Chair, (660)863-9481, carol.spradling@gmail.com, 760 W 46th St, Apt 208, Kansas City, MO 64112.
Megan Thomas, Membership System Administrator, (209)667-3584, mthomas@cs.csustan.edu, Dept. of Computer Science, CSU Stanislaus, One University Circle, Turlock, CA 95382.
Ed Lindoo, Associate Treasurer & UPE Liaison, (303)964-6385, elindoo@regis.edu, Anderson College of Business and Computing, Regis University, 3333 Regis Boulevard, Denver, CO 80221.

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

Google Cloud
GitHub
NSF – National Science Foundation

Silver Partners

zyBooks

Bronze Partners

Mercury Learning and Information
Mercy College

Welcome to the 2022 CCSC Northeastern Conference

We welcome you most enthusiastically to the Twentieth-Sixth Annual CCSC Northeastern Regional Conference at Pace University! It is so good to be able to meet together in person for the first time in three years.

The COVID pandemic has truly shaken up how we teach. Everyone has had to consider new ways of presenting material and keeping students engaged in an online or hybrid environment. While often difficult, these unusual circumstances have allowed many of us the chance to explore some of the promising new education technologies available today.

Throughout the roller-coaster ride of the past year, our CCSCNE board and conference committee members have been flexible and optimistic in their fantastic efforts to see this conference through. We are indebted to a diligent group of reviewers who have kept the conference program up to its usual standards. We are grateful for the wonderful support of Pace faculty, staff, and student volunteers in putting on this event.

All of our presenters have our sincere thanks for the work they have done in distracting times, and for their willingness to share it with us. There were 10 papers submitted, out of which 6 were accepted. This represents an acceptance rate of 60%. This year's program also includes works previously accepted at recent canceled or virtualized conferences. We look forward to our invited speaker, Amanda Holland-Minkley, and the opportunity to explore her work further in one of the conference sessions.

Finally, we thank everyone who has come to celebrate computer science education in person with us at this conference. We hope you find the experience rewarding, and that you come away with renewed enthusiasm for our field and your unique contributions to it.

Rick Kline
Pace University
CCSCNE 2022 Conference Co-Chairs
Lawrence D'Antonio
Ramapo College of New Jersey
CCSCNE 2022 Conference Co-Chairs

2022 CCSC Northeastern Conference Steering Committee

Lawrence D'Antonio, Conference Chair ..... Ramapo College of New Jersey
Richard Kline, Conference Chair ..... Pace University
Jim Teresco, Program Chair ..... Siena College
Bonnie MacKellar, Papers Chair ..... St. Johns University
Yana Kortsarts, Papers Chair ..... Widener University
Susan Imberman, Lightning Talks Chair ..... City University of New York
Susan Imberman, Panels Chair ..... City University of New York
Joan DeBello, Tutorials and Workshops Chair ..... St. John's University
Ting Liu, Tutorials and Workshops Chair ..... Siena College
Dan Rogers, Faculty Posters Chair ..... SUNY Brockport
Mike Gousie, Speakers Chair ..... Wheaton College
Karl Wurst, Student Unconference Chair ..... Worcester State University
Darren Lim, Encore Chair ..... Siena College
Sandeep Mitra, Undergraduate Posters Chair ..... SUNY Brockport
Alice Fischer, Undergraduate Posters Chair ..... University of New Haven
Stefan Christov, Undergraduate Posters Chair ..... Quinnipiac University
Liberty Page, Undergraduate Posters Chair ..... University of New Haven
Adita Kulkarni, Undergraduate Posters Chair ..... SUNY Brockport
Mark Hoffman, Registration Chair ..... Quinnipiac University
Rick Kline, Registration Chair ..... Pace University
Frank Ford, Programming Contest ..... Providence College
Del Hart, Programming Contest ..... SUNY Plattsburgh
Kevin McCullen, Vendors Chair ..... SUNY Plattsburgh

Regional Board — 2022 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

Reviewers — 2022 CCSC Northeastern Conference

Chris Alvin ..... Furman University
Kailash Chandra ..... Pittsburg
Lawrence D'Antonio ..... Ramapo College
Jamie Davilla ..... UMass Amherst
Dan DiTursi ..... Siena College
Ryan Dougherty ..... United States Military Academy
Martin Gagne ..... Wheaton College
Michael Gousie ..... Wheaton College (MA)
Nadeem Hamid ..... Berry College
Delbert Hart ..... SUNY Plattsburgh
Erin Johnson ..... CodeCrew
Sotiros Kentros ..... Salem State University
Bradley Kjell ..... Central Connecticut State University
Daniel Krutz ..... Rochester Institute of Technology
Sriharsha Mallapuram ..... Plymouth State University
Robert McCloskey ..... University of Scranton
Kevin McCullen ..... SUNY Plattsburgh
Greta Pangborn ..... Saint Michael's College
Sofya Poger ..... Felician University
Stefan Robila ..... Montclair State University
Nicholas Rosasco ..... Valparaiso University
Ingrid Russell ..... University of Hartford
Marc Waldman ..... Manhattan College

Developing a Cross–Platform Mobile Course Using a Multi–Paradigm Framework*

Alisa Neeman
Mathematics and Computer Science
Muskingum University
New Concord, OH 43762
aneeman@muskingum.edu

Abstract

Mobile programming is a popular elective in the computer science curriculum. The course designer must make choices about platform, framework, and content from amongst the many capabilities the mobile platform offers. This work shows students' understanding of multiple programming language paradigms and system issues can be strengthened when using a cross–platform, multi–paradigm framework. The practical usage of theoretical constructs is a motivating factor in their learning.

1 Introduction

Mobile course designs have often involved a choice between teaching a known imperative language with a new library, or teaching a new imperative language [12, 9, 4], in order to gain native performance. Gaining platform independence has required teaching web programming [1], or teaching a new imperative language that compiles to multiple native platforms [3].

This work focuses on the design of a Mobile course using a multi–paradigm framework, React Native. The React Native library is multi–platform, with code being compiled to the native mobile platform on which it will run.

*Copyright ©2022 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

It uses a mixture of JavaScript, CSS and XML markup languages. While JavaScript supports object oriented programming, the React Native library compels the understanding of functional programming for the library's graphical user interface. Additionally, in order to use updating or remote data resources, students used the asynchronous language supports in the JavaScript language. This work will show that using a multi–paradigm library with markup, functional, object oriented, and asynchronous threading constructs in a Mobile Apps course strengthens students understanding of some core concepts across Programming Languages and Operating Systems. The paper also covers trade–offs involved in pursuing this route.

2 Background

Previous work has revealed some important lessons for Mobile Apps course design. The work shows the tradeoffs between taking time to learn concepts or libraries, versus taking time to implement apps.

For starters, teaching Mobile is challenging because the material to be covered is very broad, including web access, sensors, GPS, database, et cetera [8]. To balance the complexity, this course covered a single asynchronous sensor (GPS) and used file–based JSON (JavaScript Object Notation) data sources as a precursor of learning NoSQL queries.

Setting up the development environment is another hurdle [2]. The complexity of a Mobile Apps course can be exacerbated by attempting to cover multiple platforms. Teaching two mobile platforms can be successful, but a tradeoff on time for course content will occur [9]. Pure web development for Mobile is alternative cross–platform approach, but misses out on native performance and full features of the device [1, 3].

The goal of this work is to teach a cross–platform framework with native performance, achieving the best of both worlds. The trade–off (and benefit) for using the React Native framework, is the necessity to cover the functional programming paradigm.

Finally, a Mobile programming course should address system issues such as security and power management, but there is a tradeoff against time for practical implementation such as creating a graphical user interface (GUI) and connecting to device sensors [11]. Although Sung and Samuel posit that focusing on practical implementation is the most suitable approach for undergraduates [11], this work shows that students can achieve an understanding of some systems issues by building an app that requires user permission to access sensitive user data (geolocation, in this case), and subscribes to repeating, asynchronous sensor updates.

3 Methods

The Mobile Apps course was a first time offering using JavaScript React Native. The course was designed for undergraduates who had previously taken some web programming, but with minimal knowledge of the JavaScript language.

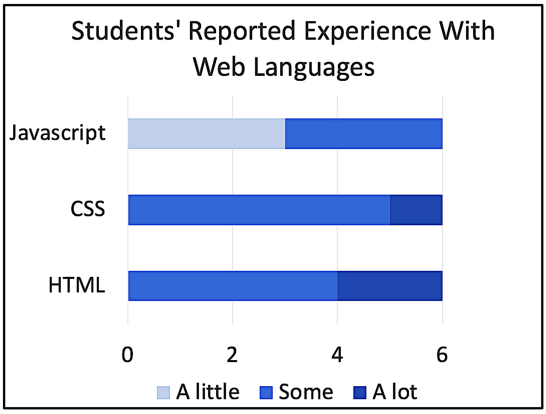


Figure 1: Students experience with web languages prior to the course.

The students reported varying levels of experience in web languages, as shown in Figure 1, and the first week of the course was focused on strengthening their basics in HTML, CSS and JavaScript. Emphasis was placed on features that would translate to React Native such as on–click action handlers, screen size–independent styling, and the Document Object Model. The remainder of the course content was ordered as follows:

- JavaScript Concepts (ES6 through JavaScript 2018)
- Setting up the environment and creating basic apps
- Creating your own React Native component, props (parameters)
- Managing state and callbacks
- CSS flexbox layout and adding images
- Stack–based screen navigation
- Maps and asynchronous geolocation tracking
- JSON files and objects

- Deployment, splash screen, and app icons
- Introduction to sound and animation

Course objectives focused on the design and implementation Mobile apps, and mobile specific issues. However, it became clear that some programming language and operating systems concepts were necessary when using the chosen framework.

3.1 Functional, Object Oriented, and Markup Features

The second week was spent covering the newest JavaScript language features, from both the Object Oriented Programming (OOP) and Functional Programming paradigms. For OOP, students learned to create both classes and JavaScript Object Notation (JSON) objects. The functional programming features were a subset of those in a typical Programming Languages class, as shown in Table 1. Arrow function syntax was unique to JavaScript. De–structured lists were declarative syntax found in JavaScript, Clojure and Python.

Table 1: Comparison of JavaScript Features and Programming Language Curriculum.

JavaScript 2018 Features	**Overlapping Functional Programming Topics**

• **First Class Functions**
• **Higher Order Functions**
• **Map**
• **Lambda Expressions**
• Lambda if-else as Map function
• Arrow Function Syntax
`const add = (x, y) =>`
`x + y;`
• **"De-structured" lists**
`[a, b] = [3, 4];`
`document.write(a); // 3`
	• **First Class Functions**
	• **Higher Order Functions** (functions that have functions as parameters)
	• **Map** (apply a function to every element of a list)
	• **Lambda Calculus**

A history of JavaScript was included so the students could understand the reason for the evolution of the language to include functional features. All of the functional language features were crucial to understanding the React Native GUI interface documentation.

Finally, students learned JSX, an XML (extensible markup language) extension to JavaScript, used to build React Native GUI components. JSX was very intuitive with their web programming background. The Composite design pattern, a pattern for a hierarchy of uniform components, was also presented [5].

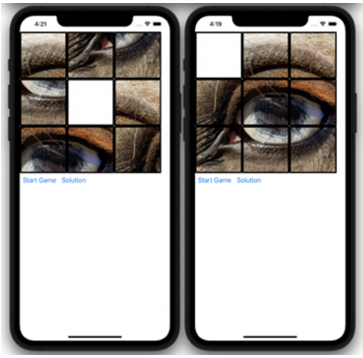


Figure 2: Puzzle App

The pattern was reflected in the XML nesting of renderable components. Understanding the relationship between JSX components and Composite design pattern, however, was not intuitive for the students, and took them some time to absorb. Then students were ready to learn to build their first app, a sliding puzzle piece game, shown in 2. The objectives were to learn file input, UI layout, screen navigation, and event handling. While assignments were individual, Scrum project management [10] was used to manage the work pace, fight fires, and get feedback on additional course content that needed to be covered. The practices included effort ranking for user stories (with discussion of the reason for the chosen rank), choice of amount of work per sprint, and daily standups where students reported what they had done, what they intended to work on, and any impediments.

Students learned about props (parameters passed in through JSX elements' attributes) and state (component–lifetime data). GUI components could be built either as a function returning a JSX component, or a class rendering a JSX component.

State was an important concept for components. The state for a React Native OOP component was initialized in a constructor, whereas a de–structured list was used for state inside a React Native component function. Table2 shows the two ways to create state for a React Native component. An interesting example of a first class function behavior emerged. React's `useState` returned a function in its de–structured list! The function was a setter for the state variable. However, state mutability meant the component was not a pure function. Interestingly, a component function definition could contain a nested function definition. A React Native function component had a structure surprisingly similar to a class, containing both state and nested function definitions.

3.2 Students Preferred Functional GUI Components

The functional programming constructs gave students something to consider, having a choice of programming paradigm. When surveyed, 5 out of 6 students preferred Functional over Object Oriented components. One student reasoned, regarding functions:

"They are easier to write because they do not require as much code to do the same thing as a class component (not using the "this" keyword all the time!).

Table 2: Two ways to create state for React Native components

OOP Style	Functional Style, using De-structured List

`constructor () {`
`  this.state = {`
`  data:initialValue,`
`  };`
`}`
	`[data, dataSetter] =`
	`React.useState(initialValue);`
	
	`dataSetter` is a function to change the `data` value.

They are also faster then class components to render, which does not matter all the time but there are times where the faster speeds are nice." [sic]

The student preferring JavaScript class components noted the similarity to Java classes used in CS1, CS2 and Algorithms:

"Using classes that represent the component as a whole helps me visualize the structure of the component better and I can tell what it is much easier. We have used classes much more often throughout, using Java in other classes [i.e. courses], that classes seem much more clean structure wise."

This shows that some students were considering programming language issues of readability, writability, and performance when given the choice of paradigms.

3.3 Using Asynchronous Threads

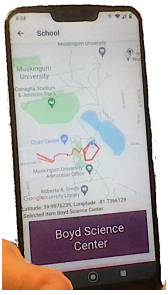


Figure 3: Map App

The second project was a geolocation tracking app to help the user find buildings on a campus map, and then find their way back to their car. Until this point, students had focused heavily on rendering their user interface, and re-rendering the UI in response to user screen interaction. The objectives of the second project were learning to use a device sensor and understanding issues associated with mobile apps.

New geolocation updates arrived after rendering the UI, and the UI could not block. Students had previous exposure to AJAX (Asynchronous JavaScript and XML) to asynchronously fetch data in the Web Programming course. For this course, they instead learned JavaScript's newer language constructs for asynchronous updates: async, await keywords, and Promise objects, which act as proxies for unknown data.

Table 3: JavaScript APIs for asnychronous data fetch

AJAX data fetch in the web browser	**Simpler `async-await` syntax for geolocation updates**

`function loadData() {`
` var x = new`
`XMLHttpRequest();`
` x.onreadystatechange =`
` function() {`
`  if (x.readyState == 4`
`  && x.status == 200) {`
 	
`document.getElementByID("demo`
`").innerHTML =`
`x.responseText;`
`  }`
` };`
` x.open("GET", "data.txt",`
`true);`
` x.send();}`
	`useEffect( () => {`
	`(`
	` async () => {`
	
	`  let location =`
	`  await`
	`Location.getCurrentPosition`
	`Async({});`
	
	`  setLocation(location);`
	`  })();`
	` },`
	`  [] );`

The new syntax was simpler to learn and more descriptive of what was happening, as shown in Table 3. Further, a real world application of asynchronous threads was a motivating experience for students.

Students' apps requested map data from a free map tile server. This worked fine in the iOS simulator, but the Android simulator simply used equivalent Google map data.

3.4 System Power Optimization and Privacy

Subscribing for continuous updates was slightly more complicated, with parameters for frequency of polling time, and minimum distance change between location updates. Without known defaults, students experimented to find the best numbers to create a nice display of the walking path on the map, while optimizing battery life. Students also learned to turn off location tracking when the app was in the background, to preserve battery life. Finally, students learned to prompt the user to permit geolocation tracking, raising their awareness of privacy when using the service.

3.5 Deployment and Platform

The only course content having distinct platform differences was in deployment. Students owned different phone platforms, so this was a concern to them. Android and iOS had separate directory structures, separate processes for compiling and bundling the app. Both deployment processes used free native platform development environments (Android Studio and Xcode, respectively) [6, 7]. The Map service was another deployment pain point. Although the apps ran well in the simulator, an Android deployment would crash unless a Google Maps license key was registered.

4 Results

Data was gathered from a class of 6 students at a small liberal arts college. Five were seniors and one was a junior. A series of surveys captured students' perceptions of their learning and likes with the course design. The surveys included Likert scale and reflection questions. Students listed what they found interesting, in their own words:

> *"Running apps inside of phone emulators."*
>
> *"The whole puzzle app in general have been enjoyable[sic]"*
>
> *"MapView and Geolocation"*

Regarding the use of Scrum practices versus a traditional, descriptive programming assignments, half had no preference, and half preferred Scrum (a third strongly preferred Scrum). Scrum worked well for managing the projects and tracking issues. It seemed to bolster student motivation, although students are generally highly motivated in Mobile courses [12, 2].

At the end of the semester, students were surveyed on whether they felt the course content enhanced their understanding of asynchronous threads, as shown in Figure 4. All students noted improvement to varying degrees.

Students reported even stronger improvement in understanding of functional programming, as shown in Figure 5.

Students were queried as to whether the Mobile course helped them understand other Computer Science course concepts. Student comments included:
"Examples of javascript code learned in mobile has been applicable to many of the lessons taught in programming languages. (and Vice Versa)[sic]"
"JSON objects and Javascript concepts (anonymous functions, map).. helped me understand Database Mgmt and Programming Languages concepts better."

When queried on what key thing(s) the students would remember from the course, answers were quite varied. One third of students listed asynchronous

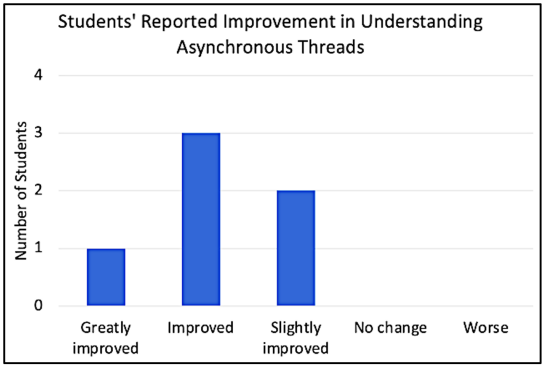


Figure 4: The degree to which understanding of asynchronous threads changed.

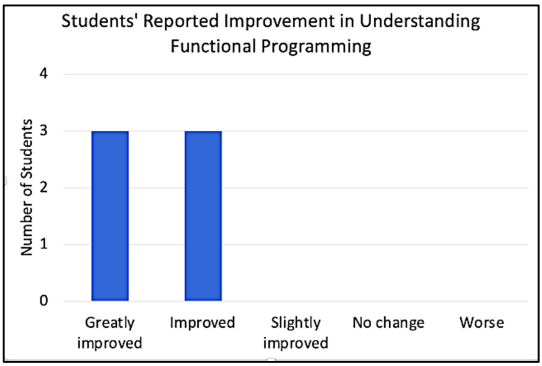


Figure 5: The degree to which understanding of functional programming changed.

threads, but props, state, arrow functions, component life cycle hooks, JSON objects, file I/O, and creating and publishing apps were listed. One student wrote that the bugs of the React Native platform would be memorable.

5 Conclusions

Teaching Mobile using React Native cross platform, multi-paradigm framework involves some trade–offs. First, it requires some background with web programming languages: HTML, CSS, and a little exposure to JavaScript. Some review will be necessary before diving deeper into JavaScript. Time must be spent to teach the constructs from functional programming, although there would be a

similar cost to teach a new imperative language for Mobile. Finally, despite the use of the framework's ability to translate to native code, deployment is still platform–dependent.

The benefits include reduced time to teach multi–platform software with native performance, as compared to teaching two separate platforms. This work clearly shows the derived benefits in strengthening student understanding of multiple paradigms: object oriented, functional, and markup, asynchronous threading, and even a new design pattern.

Scrum further enhanced to students' engagement with the course while providing important feedback for the instructor. Games and geolocation proved to be engaging assignments. Students requested further lessons on adding sound and animation to apps. Those will be good project topics for future course offerings, and both can run asynchronously.

References

[1] Peter Alston. Teaching mobile web application development: challenges faced and lessons learned. In *Proceedings of the 13th annual conference on Information technology education*, pages 239–244, 2012.

[2] Kelvin Bryant and Xiaohong Yuan. A course module on mobile programming. *Journal of Computing Sciences in Colleges*, 31(5):5–11, 2016.

[3] Paul E Dickson. Cabana: a cross-platform mobile development system. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, pages 529–534, 2012.

[4] James B Fenwick Jr, Barry L Kurtz, and Joel Hollingsworth. Teaching mobile computing and developing software to support computer science education. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 589–594, 2011.

[5] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, and Design Patterns. Elements of reusable object-oriented software. *Design Patterns. massachusetts: Addison-Wesley Publishing Company*, 1995.

[6] Google. Android studio. https://developer.android.com/studio.

[7] Apple Inc. Xcode. https://developer.apple.com/xcode/.

[8] Roy P Pargas, Punit Kulkarni, Greg Edison, and Barbara J Speziale. Teaching mobile app software development is a challenge! In *Proceedings of the 45th ACM technical symposium on Computer science education*, pages 721–721, 2014.

[9] Bryson R Payne. Teaching android and ios native mobile app development in a single semester course. *Journal of Computing Sciences in Colleges*, 30(2):176–183, 2014.

[10] Ken Schwaber and Jeff Sutherland. Scrum. https://www.scrum.org/.

[11] Kelvin Sung and Arjmand Samuel. Mobile application development classes for the mobile era. In *Proceedings of the 2014 conference on Innovation & technology in computer science education*, pages 141–146, 2014.

[12] David Wolber, Harold Abelson, and Mark Friedman. Democratizing computing with app inventor. *GetMobile: Mobile Computing and Communications*, 18(4):53–58, 2015.

Instilling Conscience about Bias and Fairness in Automated Decisions*

Sheikh Rabiul Islam[1], Ingrid Russell[2],
William Eberle[3], and Darina Dicheva[4]
[1]University of Hartford
shislam@hartford.edu
[2]University of Hartford
irussell@hartford.edu
[3]Tennessee Tech University
weberle@tntech.edu
[4]Winston-Salem State University
dichevad@wssu.edu

Abstract

Automated decision-making that impacts human interests, rights, and lives, in particular different data mining and artificial intelligence-based techniques, have become an integral part of many high-stakes applications such as sentencing and bail decisions, credit approvals, hiring, and predictive policing. However, fairness concerns, such as discrimination based on race, age, sex, etc., primarily stemming from data and algorithmic bias, is one of the major and contemporary problems associated with automated decision-making. In a traditional Data Mining, Artificial Intelligence, or Machine Learning course, educators usually teach different automated decision-making techniques but largely with limited coverage on their ethical concerns such as fairness, transparency, and

*Copyright ©2022 by the Consortium for Computing Sciences in Colleges. Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than CCSC must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

privacy. In this paper, we share our experience in building and incorporating a fairness module [1] in an existing undergraduate Data Mining course, within traditional content, and evaluate the outcome of the initiative. The module includes lectures and hands-on exercises, using state-of-the-art and open-source bias detection and mitigation software, on real-world datasets. The goal is to help instill the consciousness of fairness and bias at the very early stage of a potential future developer of automated decision-making software. The module is easily adaptable, and can be integrated into other relevant courses including introductory Artificial Intelligence and Machine Learning courses.

1 Introduction

With the advancements in computing and Artificial Intelligence-based tools and techniques, the need to analyze large and heterogeneous data has triggered automated decision-making in many real-world, high-stakes applications such as sentencing and bail decisions, credit approvals, hiring, and predictive policing. However, fairness is one of the prevailing ethical concerns with automated decision-making, and is a key stumbling block to advance the confident use of AI in application areas where the automated decisions impact human interests, rights, and lives. A few of the recent incidents of "ethical crisis" [9] in AI include Cambridge Analytica's involvement in influencing hundreds of elections globally [7], Google employee protests over military contracts [18], biased algorithms in Amazon's hiring processes [14], and racial discrimination in predictive policing [16]. In the context of decision-making, a fair decision is free from favoritism or prejudice towards individuals or groups based on their inherent or acquired characteristics. In contrast, a biased decision is skewed towards a particular person or group [13]. *Data* bias and *algorithmic* bias are the primary contributing factors for fairness related risks in AI-based decision-making. A particular group can be disproportionately represented in the *data* due to natural or systematic bias in the data collection process; and a group could be subject to statistical discrimination that is inherent in AI *algorithms*.

Traditionally, undergraduate Computer Science courses such as Data Mining, Machine Learning (ML), and Artificial Intelligence (AI) cover topics associated with automated decision-making. These courses are also core and relevant courses for a Data Science curriculum. At some institutions, these courses contain overlapping content, while at others, they are blended into one or multiple courses. However, the concept of fairness is comparatively new and is sometimes only discussed in details in graduate-level courses such as Ethics of Artificial Intelligence, and Ethics and Governance of Artificial Intelligence. In addition, at the undergraduate level, a standalone course on fairness is not

[1] https://www.dropbox.com/s/4jn88butuveaewc/bias-fairness-module-v1.pdf?dl=0

feasible due to the level of difficulty and the many other essential courses that need to be broached in a university's computer science curriculum. Therefore, instead of a standalone course, we have created a concise, high-level concept module [2] on bias and fairness in automated decisions, with (1) lecture material, (2) demonstrations, and (3) assignments (i.e., exercises) using real-world datasets. The *lecture* contains some of the prominent cases of discrimination with automated decision-making that affected human rights. For instance, in 2016, ProPublica, an independent and non-profit news provider, reported evidence of racial inequality in automated criminal risk assessment algorithms. The *demonstration* covers a bias detection and mitigation tool. For bias *detection*, we use *Aequitas*, and for bias *mitigation*, we use *IBM AI Fairness 360*. In addition, the *demonstration* is performed with a real-world case, predicting the risk score of recommitting a crime, with the well-known public dataset COMPAS. The *assignment* is related to students' term project, in this case, in a Data Mining course, which is a semester-long group project of two members per group. The *assignment* asks the students to test their term project for the *detection* and *mitigation* of potential bias and discrimination using the demonstrated tools *Aequitas* and *IBM AI Fairness 360*. Students are also asked to write their findings as a section in their group's term project report. In cases where there is no identifiable protected attribute in a project, they are told to justify, in the term project's report, why their project does not have any fairness related risks.

We included the developed module in an undergraduate Data Mining course at the University of Hartford, taught in the fall of 2020 and 2021. While the module was introduced in an undergraduate level course, it can be also introduced in graduate level AI/ML courses. Before starting the module, students were requested to take a *pre-survey* (a Likert scale survey) assessing their knowledge on fairness and discrimination. Students took the same survey again as a *post-survey*, at the end of the semester after they had completed all the components of the module. We found that the introduced module has helped to instill the consciousness of fairness and bias in students, and students found the module helpful and crucial to include in the undergraduate CS curriculum. Due to overlapping content, the module is easily adaptable and can be integrated into other relevant courses including Artificial Intelligence and Machine Learning. The initial findings is part of the poster session of SIGCSE 2022 Technical Symposium. In this paper, we describe the fairness module in Section 3. The demonstration, done with the real-world dataset COMPAS, is discussed in Section 3.2. The details of our findings are described in Section 4. In Section 5, we conclude the discussion with some future research directions. While the student module will be made publicly available, currently the lecture component of the module is available at [2].

2 Background

The integration of AI is commonplace nowadays in various aspects of human life. With the increase of various ethical concerns, it is time to revisit what future designers and developers of AI systems are learning when it comes to AI [8]. It is of paramount importance that future members of the AI community, and other stakeholders, understand and embrace their responsibilities to enhance the benefits of using AI while mitigating potential adverse effects. Towards achieving that goal, [8] emphasizes the systematic inclusion of AI ethics (e.g., privacy, fairness) into the CS curriculum, provides different approaches to AI ethics, and offers a set of recommendations related to AI ethics pedagogy.

Grosz et. al. [10] present an approach, "Embedded EthiCS", to incorporating ethical reasoning into computer science education. Their module for an introductory Machine Learning course briefly covers different theories of wrongful discrimination. However, as the module is developed for multiple courses, such as Big Data, Machine Learning, Data Systems, and Programming Languages (a few of those at the graduate level), detailed discussions of covered content, hands-on-exercises, and evaluations are unavailable.

Blair et. al. [6] propose a five-step methodology to incorporate cybersecurity concepts into traditional computer science curriculum without adding a significant amount of new content [6]. Fiesler et. al. [9] analyze 115 syllabi from university technology ethics courses that advance the inclusion of ethics in the computing curriculum. Reich et. al. [15] took a multidisciplinary approach to develop a new course by three instructors, from philosophy, political science, and computer science, at the intersection of ethics, public policy, and technology, that combines knowledge from humanities, social sciences, and computer science. They found, from students' responses, that a deeply multidisciplinary approach strongly resonates with students. According to Slavkovik et. al. [19], AI ethics is challenging to teach as the discipline itself is very new and no textbook has been established. Another challenge is introducing methodologies and skills from humanity and social sciences to CS students.

From the literature survey, it is evident that there are many prevailing ethical issues in the domain of Computer Science, in particular in the application of AI or ML which is usually the foundation of automated decision-making systems. There is no one-shot solution or perfect solution that can completely address these issues. To address various ethical issues, there is a need for multidisciplinary efforts and educational awareness.

In this paper, we share our experiences in building and using an interactive module on *fairness in automated decision-making* that engages students in detecting and mitigating bias in real-world applications.

3 Fairness Module

The module focuses on both theoretical and practical knowledge. It consists of three main components:

3.1 Lecture

The lecture component mainly focuses on the theoretical knowledge of bias and discrimination. The lecture materials [2] can be covered easily within two individual lectures, each about 50 minutes in duration, although we covered those in a single session of 75 minutes. Topics covered are as follows: (a) Various definitions of bias and discrimination, (b) Categorizations of bias and discrimination, (c) Real-world and prominent cases of discrimination with automated decision-making, (d) Fairness terms and metrics, (e) Bias detection and mitigation tools, (f) Details of the COMPAS dataset and a case study with demonstration.

The lecture starts with the definition and categorization of different biases and discrimination. According to [13], different kinds of discrimination that might occur include: (1) Direct discrimination: when the protected attributes (e.g., sex, race) of individuals explicitly result in a non-favorable outcome toward them; (2) Indirect discrimination: when rather than using protected attributes such as race, non-protected attributes such as zip code are used for decision making, but the individual can still be discriminated from the implicit effect of the protected attribute. For example, an implicit guess of the race from the zip code may lead to discrimination in a loan approval decision; (3) Systemic discrimination: results from flawed policies, custom, or behaviors (i.e., perpetuating discrimination against certain groups) that are part of the culture or structure of an organization; and (4) Statistical discrimination: results from the use of group statistics to judge an individual belonging to that group.

We then discuss the system COMPAS (Correctional Offender Management Profiling for Alternative Sanctions), which predicts the risk of a person to recommit another crime (i.e., recidivism score), and is one of the early high stakes applications of AI that garnered public attention [13]. A report reveals that COMPAS assigns a higher risk score to African-Americans than Caucasians with the same profile [12]. According to [16], a predictive policing system deployed in Chicago and New Orleans to forecast criminal activity was built on systematically biased data (a.k.a. "dirty data") resulting from unlawful practices and policies (a.k.a. "dirty policing").

We also discuss other real-world cases. For instance, a recent report reveals that native Hawaiians and Pacific Islanders are underrepresented in the census, due to different systematic barriers and a reluctance to be counted, resulting in a disproportionate allocation of resources (e.g., public funds) [4]. In addition, the ongoing COVID-19 pandemic has unmasked significant and longstanding

racial and ethnic health-related disparities in the U.S., evidenced by higher rates of COVID-19 hospitalizations, deaths, or positive cases among blacks, Hispanic/Latinos, and Native Americans at local and national levels [11]. It still remains a challenge to escape the legacy of bias and discrimination in any automated decision-making on the data reflecting systemic bias (e.g. census data).

Furthermore, we discuss some progress in assessing the fairness measures available in a tool. Some mentionable tools include: (1) *Aequitas* [17], which enables testing of different fairness metrics among different population subgroups, (2) *AI Fairness 360* [5], which is an open-source toolkit developed by IBM to examine over 70 fairness metrics and ten state-of-the-art bias mitigation algorithms developed by the research community, and (3) Google's *Fairness Indicators*, which enables the easy computation of commonly-identified fairness metrics for binary and multi-class classifiers [3]. While we resort to *Aequitas* for the rich functionalities and usability, any of the similar tools will suffice.

3.2 Demonstration

Aequitas is an open-source toolkit which can be used to audit bias and discrimination in automated decisions (e.g., Machine Learning or Artificial Intelligence based decisions), to make informed and equitable decisions. It helps to understand where biases exist in the model, compares the level of bias between protected groups, and visualizes different fairness metrics and related disparities. One has to determine fairness in relation to a *reference group*, and by default, *Aequitas* uses the *majority group* (e.g., male, white), among different attribute values of a particular attribute (e.g., sex, race), as the *reference group*.

In 2016, ProPublica, an independent and non-profit news provider, reported the evidence of racial inequality in automated criminal risk assessment algorithms [12]. Within the United States criminal justice system, COMPAS is one of the most widely utilized risk assessment tools to make a decision, such as how to set bail. It gives a risk score referring to a person's likelihood of committing a crime in the next two years. ProPublica used the COMPAS predictions data from Broward County, Florida, for the investigation, and found that, from the comparison of prediction versus ground truth, the average risk scores are higher for black defendants compared to white defendants (.51 compared to .39).

Aequitas is accessible via a Python Application Programming Interface (API), Command Line Interface (CLI), and through a web application[2] [1]. We demonstrated some of its major functions to students using the COMPAS dataset. The pipeline starts with uploading the dataset on the *Aequitas* web interface, then selecting the protected attributes in the data to consider for a

[2]http://aequitas.dssg.io

fairness test (e.g., age, sex, race), then selecting the interested fairness metrics from the available fairness metrics set, and finally, the process ends with summarized and detailed reports.

While *Aequitas* helps to *detect* bias and discrimination, it does not provide any bias *mitigation* techniques. Hence, we introduce *IBM AI Fairness 360* to the students, a similar tool but from a different vendor, for demonstrating bias *mitigation* techniques. This tool uses different techniques to mitigate bias, for example, instance reweighing which weighs the examples in each group differently to ensure fairness before classification.

3.3 Assignment

The goal of the assignment in the proposed module is to provide students with hands-on experience in assessing an automated decision-making project for possible bias and discrimination. Since we created the module for the Data Mining course at the University of Hartford, we used the course term project for that purpose. The term project is a semester-long project that requires students to work in groups of two on a data mining project of their choice using real-world datasets. The assignment asked the groups to assess their automated decision-making projects for possible bias and discrimination. Students were told to use the *Aequitas* and *IBM AI Fairness 360* tool for that purpose, and to write a section on the findings in their term paper. Some of the projects did not have any protected attributes or any possible fairness related risks. In those cases, they were told to write a section in the term paper justifying why their project does not have any fairness related risks.

4 Findings

4.1 Student Composition

The students took a survey before being introduced to the module (i.e., *pre-survey*), and another survey at the end of the semester (i.e., *post-survey*). While our class size cap is 20 (or a maximum of 25, at the instructor's discretion), for the Data Mining course, the class size was 23. Table 1 exhibits the details of the student composition including students' academic year statistics for both 2020 and 2021.

Table 1: Student Composition

Criteria	Pre-survey (2020)	Post-survey (2020)	Pre-survey (2021)	Post-survey (2021)
Class size	23	23	23	23
Total participants	21	17	20	22
Male	14	11	14	16
Female	7	6	6	5
Sophomore	1	0	0	0
Junior	6	5	10	11
Senior	14	13	10	11

4.2 Student Survey Results

The pre- and post-survey consist of eight questions. In the *post-survey* of Fall 2020, 100% of participants either strongly agreed or agreed that *"Automated decision-making can be biased towards a particular race or gender"*, whereas the percentage was 80.96 before introducing the module to students (*pre-survey*) (see Fig. 1). Similarly, the percentages for pre- and post-survey were 80 and 100 in Fall 2021 (see Fig. 2).

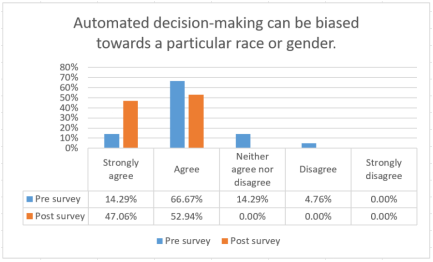


Figure 1: Race or gender bias (2020).

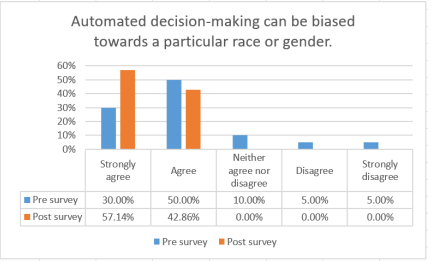


Figure 2: Race or gender bias (2021).

Similarly, in the *post-survey* of Fall 2020, 94.12% of participants either strongly agreed or agreed that *"Racial bias might be reflected in the collected data"*, whereas the percentage was 80.95 before introducing the module to students (*pre-survey*) (see Fig. 3).The percentages for pre- and post-survey were 85 and 100 in Fall 2021 (see Fig. 4).

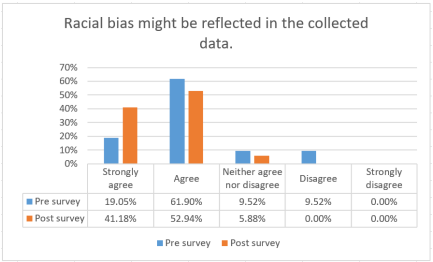


Figure 3: Data bias (2020).

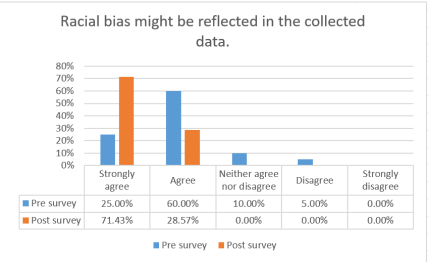


Figure 4: Data bias (2021).

Similarly, in the *post-survey* of Fall 2020, 100% of participants either strongly agreed or agreed to the question *"I think involving bias/fairness related contents in the course could help (has helped) in instilling a conscience of fairness (i.e., equity) in our minds"*, whereas the percentage was 90.47 before introducing the module to students (*pre-survey*) (see Fig. 5). Similarly, the percentages for pre- and post-survey were 95.24 and 90.47 in Fall 2021 (see Fig. 6).

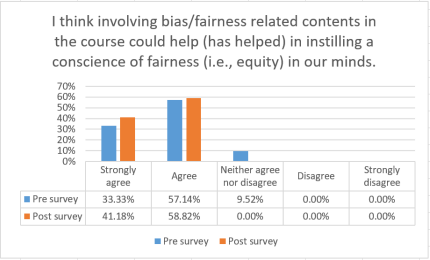


Figure 5: Instilling conscience of fairness (2020).

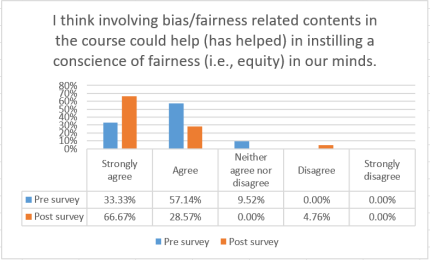


Figure 6: Instilling conscience of fairness(2021).

Overall, we see a positive impact of the proposed module on the answers of all questions in the post-survey compared to the pre-survey. So, in summary, we can say that students believed that there could be algorithmic or data bias, and the module helped to instill a conscience of bias and fairness in their minds, as evidenced by the survey results.

5 Conclusion and Future Work

Racial discrimination and disparities among different groups, such as people of different age and sex, are some of the prevailing problems in society. Because automated decisions are continuing to impact our lives, interests, and rights more and more, educating the current generation about the existence of bias and discrimination and associated implications at an early stage is crucial. The developed curricular module has the potential, evidenced by the survey, to advance this initiative. In the coming semesters, besides the Data Mining course, we plan to introduce the module to other relevant courses such as Artificial Intelligence and Machine Learning so as to cover a broader group of students. Going forward, we also plan to introduce this module at other institutions, collect and analyze student responses. Enhancements to the module will be made based on instructor and student feedback and on the latest developments in the field of fairness in automated decision-making.

6 Acknowledgments

Research reported in this work was supported by the *University of Hartford*, from the internal program "Grants to Promote Diversity, Equity, and Inclusion within the Classroom".

References

[1] Aequitas - the bias report. `http://aequitas.dssg.io/`, 2021. (Accessed on 5/18/2021).

[2] Fairness module. `https://www.dropbox.com/s/4jn88butuveaewc/bias-fairness-module-v1.pdf?dl=0`, 2021. (Accessed on 05/18/2021).

[3] Responsible ai toolkit | tensorflow. `https://www.tensorflow.org/responsible_ai/fairness_indicators/guide`, 2021. (Accessed on 05/18/2021).

[4] Gabriella Abdul-Hakim and MaryAlice Parks. *Pandemic shows need for Native Hawaiians, Pacific Islanders participation in census - ABC News*, May 2020 (accessed May 29, 2020).

[5] Rachel KE Bellamy, Kuntal Dey, Michael Hind, Samuel C Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilovic, et al. Ai fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias. *arXiv preprint arXiv:1810.01943*, 2018.

[6] Jean RS Blair, Christa M Chewar, Rajendra K Raj, and Edward Sobiesk. Infusing principles and practices for secure computing throughout an undergraduate computer science curriculum. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, pages 82–88, 2020.

[7] T Booth. Cambridge analytica controversy must spur researchers to update data ethics. *Nature*, 555(7698):559–560, 2018.

[8] Jason Borenstein and Ayanna Howard. Emerging challenges in ai and the need for ai ethics education. *AI and Ethics*, pages 1–5, 2020.

[9] Casey Fiesler, Natalie Garrett, and Nathan Beard. What do we teach when we teach tech ethics? a syllabi analysis. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pages 289–295, 2020.

[10] Barbara J Grosz, David Gray Grant, Kate Vredenburgh, Jeff Behrends, Lily Hu, Alison Simmons, and Jim Waldo. Embedded ethics: integrating ethics across cs education. *Communications of the ACM*, 62(8):54–61, 2019.

[11] Norrisa Haynes, Lisa A Cooper, and Michelle A Albert. At the heart of the matter: unmasking and addressing covid-19's toll on diverse populations. *Circulation*, 2020.

[12] Surya Mattu Julia Angwin, Jeff Larson and ProPublica Lauren Kirchner. *Machine Bias*, (accessed June 14, 2020).

[13] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *arXiv preprint arXiv:1908.09635*, 2019.

[14] David Meyer. Amazon reportedly killed an ai recruitment system because it couldn't stop the tool from discriminating against women. *Fortune. Tillgänglig online: https://fortune. com/2018/10/10/amazon-ai-recruitment-bias-women-sexist/(2019-09-27)*, 2018.

[15] Rob Reich, Mehran Sahami, Jeremy M Weinstein, and Hilary Cohen. Teaching computer ethics: A deeply multidisciplinary approach. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pages 296–302, 2020.

[16] Rashida Richardson, Jason Schultz, and Kate Crawford. Dirty data, bad predictions: How civil rights violations impact police data, predictive policing systems, and justice. *New York University Law Review Online, Forthcoming*, 2019.

[17] Pedro Saleiro, Benedict Kuester, Loren Hinkson, Jesse London, Abby Stevens, Ari Anisfeld, Kit T Rodolfa, and Rayid Ghani. Aequitas: A bias and fairness audit toolkit. *arXiv preprint arXiv:1811.05577*, 2018.

[18] Scott Shane and Daisuke Wakabayashi. 'the business of war': Google employees protest work for the pentagon. *The New York Times*, 4(04), 2018.

[19] Marija Slavkovik. Teaching ai ethics: Observations and challenges. In *Norsk IKT-konferanse for forskning og utdanning*, number 4, 2020.

An Online Tool for Easy-to-set-up, Visualizer-based, and Auto-gradable Full Tracing Exercises*

Wei Jin, David Marshall, and Puen Xie
Department of Information Technology
Georgia Gwinnett College
Lawrenceville, GA 30043
wjin@ggc.edu

Abstract

Tracing exercises are important tools to help students learn programming concepts. One common format is for students to determine the final result of a code segment. However, this often lacks specific feedback as to what misconceptions have led to a wrong answer. Full tracing exercises track and check students' tracing of a program line by line. Such exercises quickly pinpoint wrong steps and help identify misconceptions. At present, however, full tracing exercises are either difficult to set up or time-consuming to grade. As a result, they are used only sparsely in teaching. This paper describes *TracingQuiz*, a system that enables easy-to-set-up and automatic grading of full tracing exercises. First, a tracing tool was built on top of a popular online program visualizer *pythontutor.com*. It automatically injects quizzing questions for each step of execution. Only after students provide correct answers will the tool visualize a step graphically. The added benefit from the visualization feature is a more intuitive learning process. Then a mini course management system was built to make it simple for instructors to set up visualizer-based tracing assignments. Preliminary evaluation results

*Copyright ©2022 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

show that over 80% of students surveyed regard visualizer-based full tracing exercises more conducive to learning than full tracing exercises in the traditional auto-gradable formats.

1 Introduction

It is well-known among computer science educators that introductory programming courses are challenging for students [1, 8]. From our observations, many students struggle to create correct mental models of how programming constructs are executed. To help students with this, an approach used by many instructors is to demonstrate the execution of a program step by step. The demo can be done on a whiteboard or using an online tool, such as *pythontutor.com*, a popular program visualizer. A visualizer not only shows the flow of execution, but also displays graphically what each line of code does in memory or in output.

After an instructor's step-by-step demonstration, ideally students should follow up with step-by-step full tracing exercises demonstrating what each line of code does. Hopefully this process will help them identify any misconceptions they have. Some instructors ask students to "draw" what each line of code does on paper. Such exercises are only used sparsely since they are inherently time-consuming to grade. Instead, auto-gradable tracing exercises, where students determine the final result of a segment of code, are often used. Many question banks contain such questions, but these questions lack the opportunity for feedback. For a wrong answer, the system simply marks it as such without telling the student where in the tracing process they made a mistake.

Some instructors take time to set up full tracing exercises in the traditional auto-gradable formats, such as fill-in-the-blank or multiple-choice. It takes a lot of effort and time to set up exercises in a way that does not reveal the program flow and does not mention what each step does. For example, if line 6 of a program is to be executed next and it changes variable x's value to 50, the question cannot mention line 6 and cannot directly ask what the new value for variable x is; otherwise it would reveal the program flow and hint at what that line of code does. These constraints make creating full tracing exercises in the traditional auto-gradable formats tedious and time-consuming. For relatively complicated programs (e.g., programs with if statements or loops), the flow possibilities and number of steps to trace could be prohibitively high for instructors to set up questions for all steps. As a result, such exercises often have to be a scaled-down version of full tracing exercises.

To make it possible for full tracing exercises to be used more extensively, we built a new tool, *TracingQuiz*, by augmenting the online program visualizer *pythontutor.com* into an auto-gradable program tracing tool. It requires

a student to first determine which line is executed next and then answer a sequence of auto-generated questions about what that line of code does. Only if the student correctly answers the questions will the system visualize that step graphically. The tool assigns a grade for a tracing activity by subtracting points for mistakes. We also developed a mini course management system that makes it very easy for instructors to set up and deliver full tracing exercises. The visualization feature provides a more intuitive learning process, since mental models of how programs execute are often graphical in nature.

The remainder of the paper is organized in the following sections: related work, the full program tracing tool, the mini course management system, preliminary evaluation results, and conclusion and future work.

2 Related Work

Tracing exercises are often used to help students develop accurate mental models of how different constructs execute. Vainio and Sajaniemi showed that many students struggle with these exercises due to fragile understanding of program semantics [7]. Xie et. al. used a light-weight tracing strategy for paper-based line-by-line tracing [9]. It helped students score higher on both tracing problems and the course midterm exam. Hertz and Jump's approach that centered around instructor tracing demos and student tracing exercises led to statistically significant improvements in student grades, decreased drop and failure rates, and an improvement in students' programming abilities [2]. To enable wider use of tracing exercises, Kumar used the template-based technique [4] and Thomas et. al. used the stochastic tree-based technique [6] to automatically generate multiple-choice or short-answer tracing questions.

Visualization-based program tracing tools have also been proposed and proven effective for student learning [3, 5]. Kollmansberger developed a tool that reads each exercise in the XML format and renders the exercise graphically [3]. The use of this tool led to a 40% increase in course completion. Instructors could compose a tracing exercise using the XML format, but it could be a tedious process. In addition, the tool could not handle arrays, objects, and recursion.

3 Visualizer-based Program Tracing Tool

Since the program tracing tool is built on top of the program visualizer *pythontutor.com*, this section starts with the description of the base system. The visualizer first lets a user enter a complete program in a text editor. After the user presses the "Visualize" button, the user is presented with the interface to step through the program (see Figure 1). Below the source code box is a set

of buttons to single step forward (Next), fast forward to the end of the program (Last), roll the visualization one step backward (Prev), or rewind to the beginning (First). A red arrow points to the line to be executed next. When

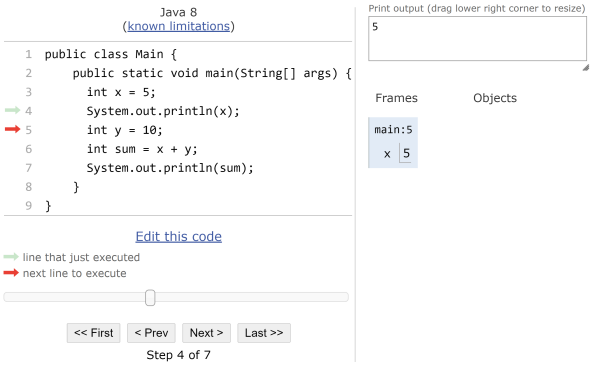


Figure 1: Pythontutor.com Visualizer Interface

the "Next" button is clicked, the result of executing the line is visualized on the right side.

- If there is a new output, it is appended to the simulated console on the top right corner of the interface. For the example in Figure 1, 5 was printed when line 4 was executed.
- Each method invocation will create a new stack frame and all the local variables will be located inside the frame. In Figure 1, there is only one frame: the frame for the main method.
- If a new variable is allocated and assigned a value, a box marked with the variable name shows up in the frame of the current method. In Figure 1, variable x was created when line 3 was executed.

TracingQuiz has exactly the same interface for a user to enter a program as that of the base system . Once the Visualize button is clicked, the system presents a modified interface as shown in Figure 2. All the navigation buttons (e.g., Next and Prev) are hidden. Users are prompted instead to click a line which they think will be executed next.

If the correct line is selected, the system presents users with a sequence of automatically generated questions. It first presents a multiple-choice question as to the action of that line of code. The choices include new output, new variable, variable update, Boolean evaluation, method call, method return, new frame, etc. When the correct action is selected, the system may present a follow-up fill-in-the-blank question, such as what the new output is, the name

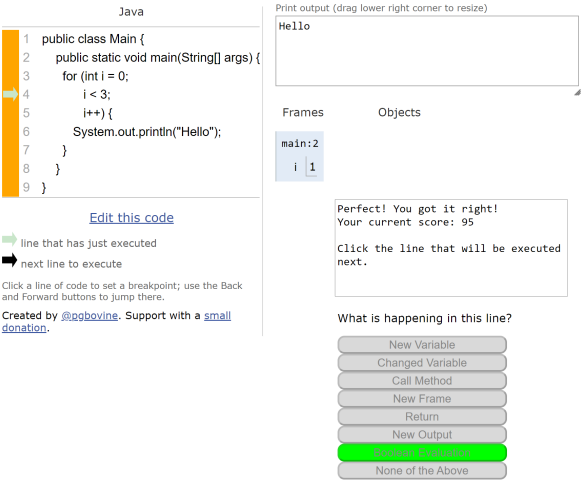


Figure 2: Program Tracing Interface

of the new/changed variable and its new value, etc. After multiple wrong attempts, the correct answer will be displayed, which prevents a student from getting stuck in one place. Each exercise starts with an initial grade of 100 points. Each incorrect answer results in points deducted.

The system handles both primitive and reference variables, as well as array indexed variables.

1. A user needs to specify the value for a variable properly according to its data type. For example, if a variable is of type double, the value has to be specified accordingly (e.g., 5.0 instead of 5).
2. For a reference variable (including an array variable), the user needs to specify the "address" of the object. However, for a new array, the user needs to enter the array content. For example, for the statement "int[] x = new int[3];", the user needs to answer "x = [0, 0, 0]".
3. For an indexed variable, the user needs to specify the indexed variable name, such as a[2].

Pythontutor.com can handle multiple programming languages, such as Java, C, C++, Python, JavaScript and Ruby. Since our introductory programming course is in Java, other programming languages are currently disabled. It will not be difficult to incorporate them in future versions of our system.

TracingQuiz can be accessed at *https://tracingquiz.xyz/visualize.html*. Students can practice full tracing exercises on programs from a typical introduc-

tory programming course. It can be accessed by anyone without a login, very similar to *pythontutor.com*.

3.1 Mini Course Management System

To make it possible for instructors to both set up program tracing assignments easily and have easy access to students' grades, we developed a mini course management system, currently deployed at *https://tracingquiz.xyz*. The system allows users to self-register. Each instructor can create multiple courses. A course can be created from scratch or by copying an existing course. A course can contain multiple sections. A student can choose their section when joining a course. The grade book also acts as the roster of the class where a teacher can assign or change a student's section.

A course contains a collection of questions and assignments (called quizzes in the system). A question is a program to be traced. All it takes to specify a question is to provide the source code. The system will check that the code has no compile errors, and will give a teacher an opportunity to trace the program before saving the question. A quiz is a tracing assignment consisting of one or more questions. Each quiz is created by picking from the existing questions.

At present, the system allows a student to work on a question as many times as they want. The highest grade for an exercise will be recorded in the grade book. Exam features will be added to the system in the future, such as only allowing students to work on a quiz for a specified maximum number of times during a specific period of time.

4 Evaluation

This section presents preliminary evaluation of *TracingQuiz* in one of the authors' sections for Programming Fundamentals in fall 2020, spring 2021 and summer 2021. Throughout the three semesters the system had been improving continuously with various bugs fixed.

Before *TracingQuiz* was built, the author set up several step-by-step full tracing exercises for the first three chapters (basics - variables and basic statements, conditionals and methods) in a learning management system, Desire2Learn (D2L). These exercises are in traditional auto-gradable formats, such as fill-in-the-blank or multiple-choice. Based on the effort undertaken to set up these exercises, we have decided to reuse the same questions across several semesters.

After *TracingQuiz* was developed, a set of tracing exercises was quickly added in the new mini course management system. Due to the convenience of creating tracing exercises in the new system, there were many more visualizer-based tracing exercises available than those in D2L.

During fall 2020, when several issues in the new system were being debugged, students used the full tracing exercises in D2L for the first part of the semester. They then moved to *TracingQuiz* when it was stable enough. Therefore, they were able to experience the difference between the two formats. The surveys indicated that students overwhelmingly preferred the new visualizer format. The evaluation process was repeated for the next two semesters (spring 2021 and summer 2021) to determine whether the same pattern would be observed. Note that all the sections were taught by the same instructor.

4.1 Comparison by Semester

Students' surveys were collected at the end of each semester. Figure 3 shows student ratings for the usefulness of in-class tracing demos, the full tracing exercises in the traditional formats in D2L (marked as D2L for simplicity), and the visualizer-based full tracing exercises in *TracingQuiz*. The possible responses range from 1-7, with 7 being Extremely Useful, 4 Neutral, and 1 Extremely Useless.

The average ratings for instructor's tracing demos are above 6 for all semesters. For the visualizer-based exercises, the rating increases with each new semester, as the system becomes more stable and more tracing exercises are added. The average rating for the visualizer format in *TracingQuiz* is higher than that of the traditional formats in D2L for each of the semesters. None of the differences, however, are statistically significant. Note that the significance tests mentioned in Section 4 are one-way ANOVA tests unless otherwise stated.

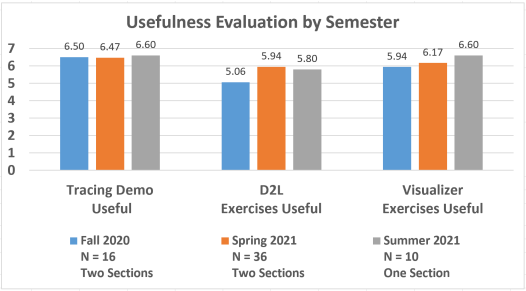


Figure 3: Usefulness of the Tool by Semester

4.2 Comparison by Efforts

Students self-identified themselves into three groups: completed all visualizer-based tracing exercises (Did All), completed more than half of those exercises

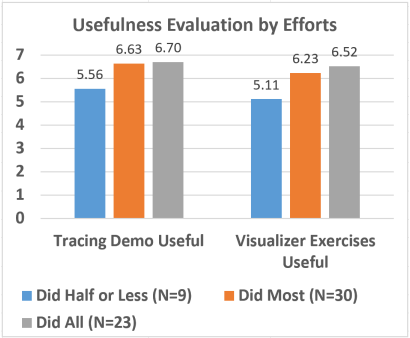


Figure 4: Evaluation of the usefulness by different groups.

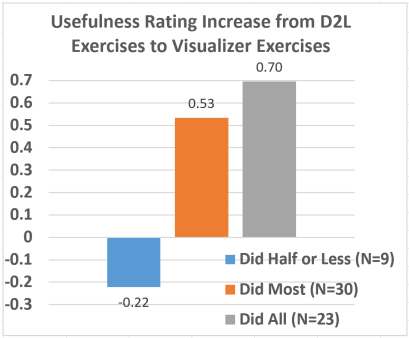


Figure 5: Average rating increases for different groups.

(Did Most), completed half or fewer of those exercises (Did Half or Less).

Figure 4 plots the average group ratings for the usefulness of tracing demos and visualizer-based exercises. All the ratings are high. There is a clear trend that the more visualizer-based exercises students did, the better they rated both, with the "Did All" group giving an average rating 6.6, where 7 is the highest rating possible. The differences among the groups are statistically significant for visualizer-based tracing exercises ($F(2, 59) = 4.585$, $p = .014$) as well as for tracing demos ($F(2, 59) = 13.886$, $p = .000$). Turkey post hoc tests show that the "Did Half or Less" group has statistically significant difference from both the "Did Most" group and the "Did All" group for both exercises and demos.

The usefulness rating increase from the traditional formats to the visualizer format in *TracingQuiz* was calculated for each student and then averaged for each group. Figure 5 shows a clear increasing pattern from "Did Half or Less" group to the "Did All" group. For the "Did All" group, the average increase is around 0.7, i.e. 70% of one level up. A paired samples t-test for all samples shows that the difference between the traditional formats and the visualizer format is statistically significant ($p = 0.012$). Note that the differences among different groups might be correlation – students with early difficulties in completing the exercises may have given upon trying more.

To gain more insights, students were asked to compare the two formats directly. Figure 6 shows students' opinion about whether all tracing exercises should be in the new format. All groups leaned towards the new format. In fact, 82.3% of all students said that all tracing exercises should be or probably should be in the new format, with 43.5% definitely yes and 38.7% probably yes. A similar trend repeats here: the more effort, the more affirmative answers.

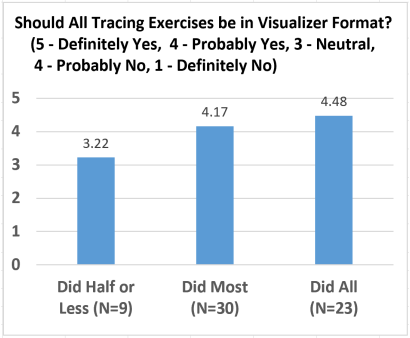


Figure 6: Groups' recommended use of tracing exercises.

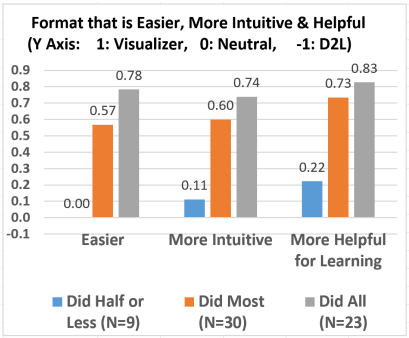


Figure 7: Groups' ratings for which format is more helpful.

Figure 7 shows which format students regarded as easier to use, more intuitive, and more helpful for learning. All groups again favored the visualizer format, except that the "Did Half or Less" group was neutral in the easiness measure. In fact, 74.2%, 75.8%, and 80.6% of all students felt that the visualizer format is the choice for the three measures, respectively. Again, the more exercises students did in the visualizer format, the more they favored it.

5 Conclusion and Future Work

Surveys show that students overwhelmingly preferred *TracingQuiz* for full tracing exercises. 80.6% of students regarded the new format as more helpful for learning. Anecdotally, quite a few students commented that they realized the value of these exercises after completing several assignments. The data also show that the more visualizer-based tracing exercises students completed, the more students believed that they were conducive to learning.

Students' evaluations of the system improved over the three semesters as the system gradually improved. The system will be continually improved for wider adoption by the teaching community. The following are some planned new features: ability to trace more programming constructs (e.g., multi-dimensional arrays, array of objects, and inheritance), tracing exam features in the mini course management system, and integration of our system with many widely used learning management systems, such as Canvas and D2L.

The ease of creating full tracing exercises in *TracingQuiz* will hopefully enable instructors to focus more on creating quality exercises to facilitate student learning. Several instructors will join the effort in spring 2022 to examine the current tracing exercises and improve the collection so that they adequately

address tricky or difficult concepts for each topic, such as the difference between multi-branch if statements and multiple separate if statements, the difference between primitive variables and reference variables, the difference between static methods and instance methods in terms of the implicit parameter for instance methods, etc. Furthermore, analysis of student interactions (currently not logged) with the system will shed light on the concepts that are difficult and confusing for students. This could help narrow down when and where tracing code is necessary as well as develop better tracing exercises.

On the evaluation front, the plan is to increase the number of sections and students participating in the study. In addition to attitudinal survey evaluations, evaluation for whether the tool actually helps improve student learning will also be conducted.

References

[1] Jens Bennedsen and Michael E Caspersen. "Failure rates in introductory programming". In: *AcM SIGcSE Bulletin* 39.2 (2007), pp. 32–36.

[2] Matthew Hertz and Maria Jump. "Trace-based teaching in early programming courses". In: *Proceeding of the 44th ACM technical symposium on Computer science education*. 2013, pp. 561–566.

[3] Steven Kollmansberger. "Helping students build a mental model of computation". In: *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education*. 2010, pp. 128–131.

[4] Amruth Kumar. "Dynamically generating problems on static scope". In: *ACM SIGCSE Bulletin* 32.3 (2000), pp. 9–12.

[5] Juha Sorva and Teemu Sirkiä. "UUhistle: a software tool for visual program simulation". In: *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*. 2010, pp. 49–54.

[6] Anderson Thomas et al. "Stochastic tree-based generation of program-tracing practice questions". In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. 2019, pp. 91–97.

[7] Vesa Vainio and Jorma Sajaniemi. "Factors in novice programmers' poor tracing skills". In: *ACM SIGCSE Bulletin* 39.3 (2007), pp. 236–240.

[8] Christopher Watson and Frederick WB Li. "Failure rates in introductory programming revisited". In: *Proceedings of the 2014 conference on Innovation & technology in computer science education*. 2014, pp. 39–44.

[9] Benjamin Xie, Greg L Nelson, and Andrew J Ko. "An explicit strategy to scaffold novice program tracing". In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. 2018, pp. 344–349.

Content-Synchronized Game Development Modules in CS1*

Xin Xu, Wei Jin, Hyesung Park,Evelyn Brannock, Adrian Heinz
Information Technology
Georgia Gwinnett College
Lawrenceville , GA 30043
{xxu,wjin,hpark7,ebrannoc,aheinz}@ggc.edu

Abstract

Development of nontrivial games is normally reserved for upper level courses after students have gained adequate knowledge and skills. This paper examines the incorporation of development of popular games (resembling Flappy Bird, Snake, TRex, etc.) into an introductory programming course (CS1). A sequence of modules that students follow to build a game was developed, with each module focusing on one programming topic. Effort was expended to ensure that the content order of a typical CS1 course was retained within the modules. This synchronization hopefully helps students see the relevance of the programming concepts in real-world applications and improve interest and motivation in learning. The initial survey results are promising. 86% of the students would like to have more of this type of workshops. In addition, students' curiosity and interest in programming increased, especially among female and African American students.

1 Introduction

Many studies have shown that students in introductory programming courses have difficulty learning new concepts and writing code[6, 14]. Students often perceive programming to be dry, labor-intensive, or time-consuming. These

*Copyright ©2022 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

perceptions can make them lose interest or motivation to continue learning[1]. Students get easily frustrated when they cannot solve a programming problem, and quickly lose enthusiasm to continue trying. These challenges are reflected in past studies[5, 16], which found that the average pass rate in introductory programming courses is a disappointing 67%.

This paper describes an approach that utilizes course-embedded game development modules to improve student interest and curiosity in programming. Development of nontrivial games is normally reserved for upper level courses after students have gained adequate knowledge and skills. Our approach is to incorporate some advanced game development seamlessly into an introductory course. Games, such as Flappy Bird, Snake, TRex, and Circle Dodge, are chosen due to their popularity among students. Familiarity with these games will help draw students in and improve their curiosity.

Each game is divided into a sequence of carefully designed course modules. The process of developing a game is utilized as a way to introduce some basic programming concepts, help students see the relevance of what they are learning, and increase their interest in programming. Each module focuses on a specific programming topic and the order of the topics is roughly the same as that in a typical CS1 course. Dividing a big task into smaller subtasks makes the workload manageable. Completing a game one small step at a time gives students a sense of accomplishment along the way, which improves self-efficacy and motivation.

We also leverage peer modeling to motivate students. A group of students who recently completed the introductory programming course were recruited to develop games and course modules and eventually conduct the game development workshops. Processing[7] was chosen as the platform because it is open source, based on Java (the language of our CS1 course), and easy to learn.

This project was implemented in multiple sections of the CS1 course in Spring 2021. The results are very encouraging. Students showed improved attitude toward programming, improved curiosity and motivation for learning. The rest of this paper will present the related work, the project details, the evaluation results, and finally, the conclusion and future work.

2 Related Work

Game development improves student motivation and engagement, resulting in the increased attraction of new students[2, 3, 12]. Bayliss and Strout[3] and Luxton-Reilly and Denny[13] used the strategy of assigning game-themed homework around the technical topics and reported positive results in achieving the learning outcomes. Authors of [15] studied the impact of students' learning outcomes through scenario-based active learning with Unity 3D. The

study found that the active learning method with game development helped students understand different subject concepts. Students also valued game-based learning and experienced improved learning effectiveness, motivation, and persistence.

Golding et al. studied the impact of peer modeling and student attitude, confidence, and academic performance[9]. Chase and Okie reported that the DFW rate, especially for female students, dropped based on peer modeling as part of cooperative learning[8]. Many studies showed that peer modeling improved student motivation and engagement and increased retention[4, 6, 11, 16].

Some studies have also shown that learning different programming platforms in one class produces better results[10]. In this study, Processing was an additional tool in an introductory programming class. The Processing development environment provides an easy-to-use interface, and the Processing language is based on Java and easy to learn. Starting with simple and easy steps, students can develop a complex application relatively quickly.

3 Project Description

In fall 2019 and spring 2020, several IT students who recently completed CS1 were hired to develop Processing games and then course-embedded workshop modules based on these games. To make the workshops engaging, students chose to develop popular games among young people, including Flappy Bird, Snake, TRex, and Circle Dodge. Student developers then worked together with faculty mentors and divided the game development of each game into a sequence of stages. Each stage completes part of the game and focuses on one main programming topic. The order of the topics covered by these stages roughly matches the order that these topics are usually taught in a CS1 course. A workshop module was developed for each stage and it consists of the following components:

- Starter code
- A PowerPoint file with detailed step-by-step instructions
- Solution code
- A description of the follow-up homework

The objective was that students would build a sense of accomplishment through this multi-module approach instead of being overwhelmed by the complexity of the whole project. The homework after the workshop was designed to foster critical thinking and problem-solving skills. The solution of the homework is also the starter code for the next workshop, allowing students without a perfect solution to proceed further without feeling left behind.

The original plan was to have the student developers present the workshops in class. When piloting the project in spring 2020, one workshop was conducted by student developers in person. The rest had to be conducted online because of COVID. Despite the unexpected situation, positive feedback was received for the pilot study.

In fall 2020, the project was promoted among instructors teaching CS1. Four sets of game workshop modules based on four different games were shared, and the strategies for embedding these workshops in the course were discussed. Four faculty members decided to adopt the game workshop modules in their sections in spring 2021. The evaluation results presented below are based on the workshops conducted by faculty members in the online synchronous format.

4 Evaluation of Impact on Student Motivation

This paper focuses on the project's impact on student motivation. The evaluation results provide valuable insight on how to further improve the project. Evaluation for impact on student learning will be included in the future work.

At the end of spring 2021, students were asked to complete a survey after each game development workshop. A total of 159 responses were received. Survey results show that the student body is quite diverse with 30% African American, 22% Asian, 22% White, 20% Hispanic, and 6% others. For gender distribution, 77% are male students and 18% female students.

The following are four aspects that the surveys were designed to evaluate:

- students' overall experience with these workshops
- students' attitude about a programming course
- the factors that motivate students and make them become more curious about programming
- the synchronization effect of the game workshop topics with the course topics

4.1 Analysis of Workshop Experience

Figure 1 shows the survey results for whether the students actively participated in the workshop, became more curious about programming, learned something new, etc.

As shown in the left chart of Figure 1, the averages of the responses for all the questions are above 4.0 for both the male and female group. For the question "would you like to have more of this type of workshop", 86% of all students answered with 4 (Yes) and 5 (Definitely Yes). Female students had more positive responses than male students for engagement, learning something new, and the overall experience. However, for "would you like to have more

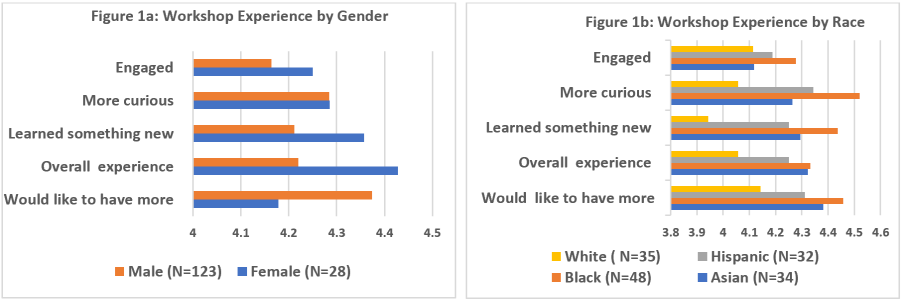


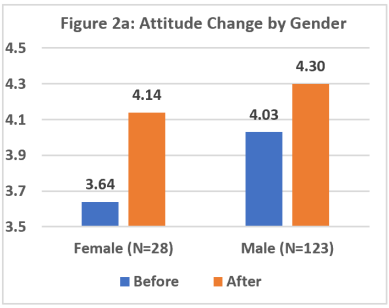
Figure 1: Analysis of Workshop Experience

of this type of workshop", male students had higher positive responses (4.37) than female students (4.17). Is it because males are more interested in games and game development in general? Or were the games used in this project more male oriented? The response prompts the need for further investigation to be conducted. For different race groups, results showed that black students had slightly more positive responses for the workshop experience (see the right chart in Figure 1)). The differences among either the gender groups or the racial/ethnic groups are, however, not statistically significant.

4.2 Analysis of Students Attitude

Students were asked to evaluate their attitudes towards programming before and after the workshop with a Likert scale from 1 to 5 with 1 indicating not interested at all and 5 indicating extremely interested. As shown in the left chart in Figure 2, there is a significant increase of interest for both female and male students but the increase is more for female student (from 3.64 to 4.14) than male students (from 4.03 to 4.30). According to a paired t-test, the change experiences by both male and female groups are statistically significant ($p < 0.0001$ for both groups). This indicates that these workshops were helpful in motivating students and changing their perception of programming, particularly for female students.

As shown in the right chart of Figure 2, these workshops motivated African American students (from 3.81 to 4.29) and Asian students (from 3.68 to 4.18) more than Hispanic students and White students (from 4.13 to 4.28 and 4.29 to 4.37 respectively). The changes for the Asian and Black students are statistically significant according to a paired t-test ($p < 0.0001$ for both groups).



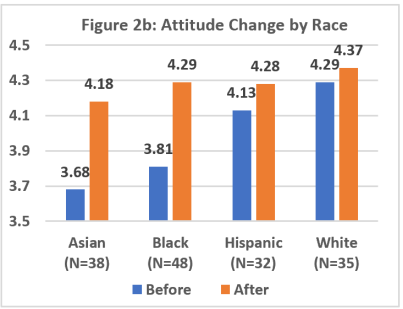


Figure 2: Change of Attitude by Gender and Race

4.3 Analysis of Motivational Factors

To investigate what factors motivated students the most, students were asked to rate the following statements with a scale from 1 to 5, with 1 indicating not effective at all and 5 indicating most effective. Feedback from these statements hopefully could provide some insight for future workshop design.

Table 1: Motivational Factors

Survey Questions	Result
It seems not too hard to create a game from scratch	4.22
The tutorial was engaging	4.14
I can be creative with programming	4.31
I'm learning a new technology	4.31
Able to apply programming concepts in a new environment	4.33
This game was developed by a peer student	4.15

Table 1 shows that all the factors received above 4.0 ratings in the scale of 1 - 5, which indicates that all of the factors played important roles in motivating students and helped them become more curious about programming. Learning a new technology, applying knowledge in a new environment, and being able to demonstrate creativity had a slightly higher impact (above 4.30) than the rest. Further analysis (Figure 3) also discovered that "the fact that these games were developed by a peer student one or two years ahead of me" had a higher impact on female students (4.3) than male students (4.1), and, "the fact that it seems not too hard to create a game from scratch" had a higher impact on male students (4.23) than female students (4.07). The difference among different racial/ethnic groups for the motivational factors, however, does not show any clear patterns.

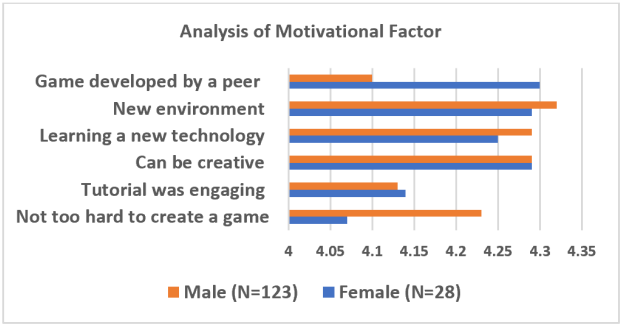


Figure 3: Motivational Factor Analysis by Gender

4.4 Analysis of Degree of Synchronization with Course Content

In spring 2021, without prior discussion and arrangement, faculty members chose two out of the four sets of workshop modules, one based on Flappy Bird and another based on the Snake game. For Flappy Bird, each workshop module was further subdivided into smaller activities better matched to the programming concepts being introduced in class. Each mini module was used to introduce students to a specific topic, such as conditional statements and methods. The finer-grained workshops made the embedded activities more synchronized with the course content. The highly synchronized approach required more time and effort for the instructor to adjust the materials to match the topics well.

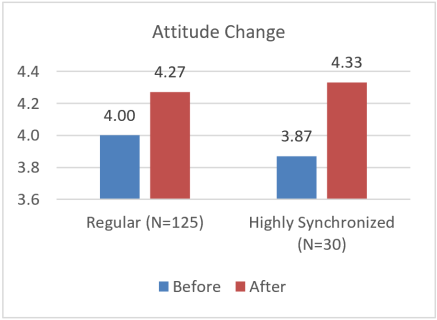


Figure 4: Attitude Analysis by Degree of Synchronization

Analysis results showed that the higher the degree of synchronization of the workshops with course content, the greater the impact is on student attitude.

As shown in Figure 4, the highly synchronized Flappy Bird workshops increased the attitude from 3.87 to 4.33, while the regular workshops only increased the attitude from 4.00 to 4.27. Paired t-test results suggest that there is a statistically significant difference in attitude before and after the workshop for both highly synchronized workshops and regular workshops but the p-value for highly synchronized workshops ($p<0.0001$) is much lower than that of regular workshops ($p=0.028$).

Students participating in the highly synchronized workshops also gave each motivating factor a higher rating than that of the regular workshops (see Figure 5). For "not too hard to create a game", the rating difference is the largest (4.5 vs 4.15) and statistically significant ($p=.047$).

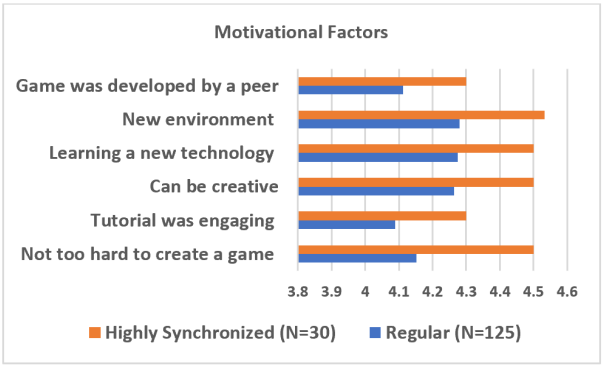


Figure 5: Motivational Factor Analysis by Degree of Synchronization

5 Conclusion and Future Work

From the analysis of a semester of data from four instructors, the preliminary results appear to be promising. Three of the motivational factors (creativity with programming, learning a new technology, and applying programming concepts in a new environment) averaged above 4.3 (out of 5) (Table 1). Female and African American students reported a better workshop experience than the other gender and racial/ethnic groups. Female, Asian, and African American students reported a more positive change of attitude towards programming after these workshops.

There are two candidate dimensions for extending the project. First, more diverse games and types of workshop modules could be developed to give instructors additional freedom of choice based on their pedagogical needs. For example, instead of a sequence of modules for one game, an instructor could

use a module that starts with an almost complete game, with only a single missing piece that matches to the current specific topic. Data presented in Section 4.1 led the investigators to examine whether the games chosen currently are male-oriented. More diverse games may shed light on what types of games might work better for female students.

Second, more data will be collected for a more rigorous study. Because our institution has a highly diverse student composition, the authors would like to gain a better understanding of various motivational factors for different demographic groups. The authors are also interested in what granularity level of the workshops could have the most constructive impact on learning. Assessment for improvement in learning will be included in the future evaluation plan.

6 Acknowledgment

The project is supported by NSF funded (NSF Award ID 1623779) Course-embedded Undergraduate Research Experiences (CURE) Mini-Grant.

References

[1] Robert L Avanzato. Collaborative mobile robot design in an introductory programming course for engineers. In *1998 Annual Conference*, pages 3–145, 1998.

[2] Jessica D Bayliss. Using games in introductory courses: tips from the trenches. In *Proceedings of the 40th ACM technical symposium on Computer science education*, pages 337–341, 2009.

[3] Jessica D Bayliss and Sean Strout. Games as a" flavor" of cs1. In *Proceedings of the 37th SIGCSE technical symposium on Computer science education*, pages 500–504, 2006.

[4] Susan Beltman and Marcel Schaeben. Institution-wide peer mentoring: Benefits for mentors. *The International Journal of the First Year in Higher Education*, 3(2):33–44, 2012.

[5] Jens Bennedsen and Michael E Caspersen. Failure rates in introductory programming. *AcM SIGcSE Bulletin*, 39(2):32–36, 2007.

[6] Gregory Bucks and William Oakes. Work in progress-impact of graphical programming environments on learning and understanding programming concepts. In *2008 38th Annual Frontiers in Education Conference*, pages F2A–23. IEEE, 2008.

[7] Ben Fry Casey Reas. Processing application. `https://processing.org/`.

[8] Joe D Chase and Edward G Okie. Combining cooperative learning and peer instruction in introductory computer science. In *Proceedings of the thirty-first SIGCSE technical symposium on Computer science education*, pages 372–376, 2000.

[9] Paul Golding, Lisa Facey-Shaw, and Vanesa Tennant. Effects of peer tutoring, attitude and personality on academic performance of first year introductory programming students. In *Proceedings. Frontiers in Education. 36th Annual Conference*, pages 7–12. IEEE, 2006.

[10] Paul Graham and Troy Weingart. Processing language in introduction to computer science honors (cs110h). *Journal of Computing Sciences in Colleges*, 25(2):72–78, 2009.

[11] Ville ISOMöTTöNEN and Vesa Lappalainen. Csi with games and an emphasis on tdd and unit testing: piling a trend upon a trend. *ACM Inroads*, 3(3):62–68, 2012.

[12] Scott Leutenegger and Jeffrey Edgington. A games first approach to teaching introductory programming. In *Proceedings of the 38th SIGCSE technical symposium on Computer science education*, pages 115–118, 2007.

[13] Andrew Luxton-Reilly and Paul Denny. A simple framework for interactive games in cs1. In *Proceedings of the 40th ACM technical symposium on Computer science education*, pages 216–220, 2009.

[14] Michael McCracken, Vicki Almstrum, Danny Diaz, Mark Guzdial, Dianne Hagan, Yifat Ben-David Kolikant, Cary Laxer, Lynda Thomas, Ian Utting, and Tadeusz Wilusz. A multi-national, multi-institutional study of assessment of programming skills of first-year cs students. In *Working group reports from ITiCSE on Innovation and technology in computer science education*, pages 125–180. 2001.

[15] Hyesung Park, Sean Yang, and Hongsik Choi. Scenario based active learning programming with unity 3d. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pages 1283–1283, 2020.

[16] Christopher Watson and Frederick WB Li. Failure rates in introductory programming revisited. In *Proceedings of the 2014 conference on Innovation & technology in computer science education*, pages 39–44, 2014.

Cybersecurity Shuffle: Using Card Magic to Teach Introductory Cybersecurity Topics*

Preston Moore[1] and Justin Cappos[2]
New York University
New York, New York, United States
pkm266@nyu.edu[1] jcappos@nyu.edu[2]

Abstract

One of the main challenges in designing lessons for an introductory information security class is how to present new technical concepts in a manner comprehensible to students with widely different backgrounds. A non-traditional approach can help students engage with the material and master these unfamiliar ideas. We have devised a series of lessons that teach important information security topics, such as social engineering, side-channel attacks, and attacks on randomness using card magic. Each lesson centers around a card trick that allows the instructor to simulate the described attack in a way that makes sense, even for those who have no prior technical background. In this paper, we describe our experience using these lessons to teach cybersecurity topics to high school students with limited computer science knowledge. Students were assessed before and after the demonstration to gauge their mastery of the material, and, while we had a very limited set of responses, the results show an improvement on post-test scores. Furthermore, several indicators affirm the students enjoyed the lessons and remained engaged throughout the session.

1 Introduction

When teaching technical topics in introductory courses, it can be challenging to present information in a way that makes sense for students of varying experi-

*Copyright ©2022 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

ence levels or educational backgrounds. This is particularly true for information security classes where an adversarial mindset is required to fully comprehend the attacks. Thinking in this way may not come naturally to many students, as evidenced by the continuing success of phishing attacks. What is needed is a way to relate information security concepts in a manner that is engaging enough to build an appreciation for the material, yet relatable enough so the students do not feel lost. To accomplish this, we look towards a pedagogical technique known as scaffolding in which "students are escorted and monitored through learning activities that function as interactive conduits to get them to the next stage [9]."'

In this paper, we employ card magic as a scaffolding device in a series of lessons that teach how three types of attacks – social engineering, side channel attacks, and attacks on randomness – work in the real world. In doing so, we add a new twist to the success other computer science researchers have had in using card magic to explain difficult concepts by allowing the instructor to simulate "attacks" through a non-technical, commonplace activity. In doing so, these lessons can help students safely interact within these attack scenarios. Each demonstration was followed by a short PowerPoint presentation in which the magician makes a connection between the card trick and the very real consequences of the attack it illustrates.

To test the effectiveness of our lessons, we presented them to a group of high school students attending a computer science summer program and assessed mastery using a pre- and post-test. Though our sample size was too small to draw definite conclusions from them, participant scores did increase on the post-test for each subject. Furthermore, based on an opinion survey and presenter observations, participants found the lessons engaging, age appropriate, and helpful in understanding the concepts.

- We create a lesson plan built around three easy-to-perform magic tricks. By using these tricks, instructors can simulate attacks and thus provide a scaffold for teaching these somewhat difficult concepts.
- We test the effectiveness of these lessons by presenting them to a group of high school students in a summer workshop and assessing engagement and improved mastery of the material.
- We note an improved ability to answer questions related to the attacks following our lesson, as judged by pre- and post-test evaluations.

2 A Lesson Wrapped in an Illusion

A magician creates an illusion to hide the secrets of his or her tricks. The lessons we have developed reverse this situation by using our tricks to reveal the mys-

teries behind three cyberattacks. The tricks were chosen because of their relevance to the information security field. Following the principles of scaffolding, the attacks are presented in order of increasing technical complexity. Video tutorials for each of the tricks are available at: *https://bitly.com/cybersecurityshuffle.* [1]

2.1 Social Engineering

In the context of information security, "social engineering" is defined as a set of tactics to manipulate users into giving away personal information that can be used to compromise accounts, reset passwords using security questions, or carry out identity theft. We start with this attack as it is broadly used and affects arguably the greatest cross-section of victims. In our lesson, the magician employs two card tricks as a misdirection and a cover to distract from the amount of personal data he/she is soliciting.

This trick is intended to spark a teachable moment about social engineering and its dangers. Instructors can use this moment to start a dialog with students about the types of information attackers might want and how they could maliciously use it.

2.1.1 From the Audience's Perspective

Our version of this trick is adapted from a magic classic known as "The Red and Black Separation Trick [8]." The magician begins by telling the audience that there is a way to form a psychic bond with a deck of cards. The magician enlists a volunteer and each shuffles the deck before placing one half on top of the other. Next, the magician asks the volunteer a few questions, starting with their birth year, to allegedly "attune" the link between the individual and the deck. The response is used to select one red card and one black card from the deck, each with a numeric value equal to one of the last two digits of the volunteer's birth year. Next, the magician asks for the volunteer's birth month and similarly selects a red card with a numeric value equal to the response (using the jack and queen for November and December, respectively). Finally, the magician asks for the volunteer's birthday and selects a black card with a numeric value equal to the second digit in this day. The magician lays these cards out on the table and asks the volunteer to select one red card and one black card with which they feel most "attuned." The unchosen cards are returned face up to the middle of the deck.

Continuing the pretense of a psychic link with the deck, the magician asks the volunteer to guess the color of each card in the deck. As he/she does so,

[1] Public domain and GPL-licensed card images used in figures taken from Wikimedia Commons [13].

the magician places each card face down in two piles: red and black. Half-way through the deck, the two face up cards are switched so the black pile becomes the red pile and vice-versa. The volunteer continues guessing until all the cards are placed. At that point the magician turns over the face down cards to reveal that the volunteer has guessed **e**very card's color correctly. The magician then reveals the "misdirection" that abetted the trick's true purpose – revealing personal information.

2.1.2 Behind the Scenes

Before the trick begins, the deck has already been separated into red and black halves. In the initial shuffling the volunteer is merely scrambling cards of the same color. Therefore, when the two packs are stacked one on top of the other, the two colors remain separate.

In the first portion of the trick the magician pulls out two red and two black cards that reflect the volunteer's answer, and two of these cards are returned to the deck. The magician must return these cards face up exactly between the red and black "sections." This will later signal the magician when all cards of one color have been dealt.

During the prediction phase of the trick, one pile contains all correct guesses while the other is completely incorrect. When the midway point is reached the magician places the red card face up on the black pile and vice versa. An illustration of this arrangement appears in Figure 1. In the reveal the magician flips the correct pile horizontally and the "incorrect" pile forward vertically to reverse the incorrect guesses. The red cards are now paired with the red marker card and vice-versa to complete the illusion that the volunteer correctly guessed **e**very card in the deck.

2.2 Side Channel Attacks

As the name implies, a side channel attack strikes a target indirectly by tracking seemingly unrelated phenomena, such as timing information, power consumption, electromagnetic leaks, or even sounds. To mirror this type of attack, the trick demonstrates how an attacker can gather information without directly exploiting a vulnerability. We do so using a deck of cards with a brand logo on the back. When turned upside-down the logo effectively creates the equivalent of a "mark," similar to a "marked" deck of cards. The goal of this trick is to open the participant's eyes to the less obvious avenues an attacker might use to transmit information.

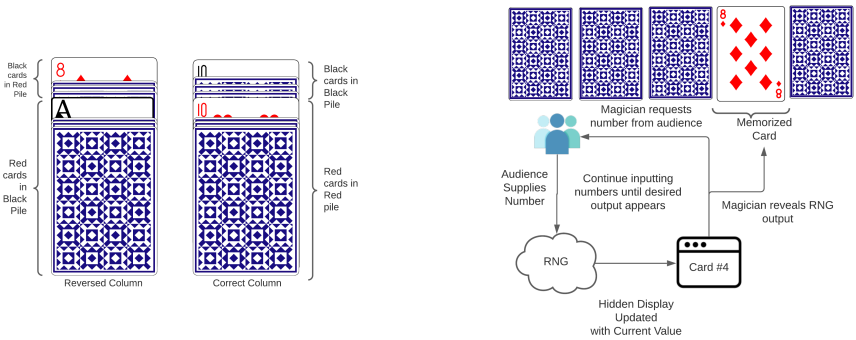


Figure 1: Left: Arrangement of cards after guessing concludes in Trick 1. Right: Use of "random number generator" in Trick 3

2.2.1 From the Audience's Perspective

The magician opens a new deck of cards, removes the jokers and branding cards, and legitimately shuffles it. Several volunteers are asked to select a card from the deck, show it to the audience, memorize it without revealing it to the magician, and then return it to the deck. The magician then shuffles the deck before going through it and finding all of the volunteers' cards. This trick's reveal comes when the magician informs the audience that the cards were found using a secret information side channel present in the deck, opening a door to explore further side channels.

2.2.2 Behind the Scenes

The key to this trick lies in the magician's choosing a deck with a logo or text on the back that tips off a card's orientation. Because the trick begins with a fresh deck, all cards are oriented in the same direction. When the volunteers return their cards to the deck, the magician simply has to orient the deck in such a way that the returned cards are upside-down. To complete the trick, the magician finds the card with a differing orientation.

2.3 Attacks on Randomness

True randomness is important in many security-sensitive situations. An attacker who is able to predict or influence the output of a random number generator may use this capability to circumvent cryptographic security controls. This trick employs a "forced" card [14] to point out a potential vulnerability that results from a misunderstanding of hash functions.

The purpose of this lesson is to show students the importance of correct randomness that is "fit for purpose," or appropriate for security-sensitive applications. It also shows how an attacker with a small amount of influence, such as when to stop supplying numbers, can compromise a system.

2.3.1 From the Audience's Perspective

The trick begins with the magician announcing that it is possible to guess the value of any randomly selected card in a deck by touch. To prove this point, the magician spreads a deck of cards on the table face up, shuffles the deck and deals five cards face down. In order to head off suspicion that the cards are "fixed," the magician declares a software random number generator will be used to select which card will be predicted. Students are asked to shout out numbers to be input into the generator. After a handful of numbers, the magician cuts off input, generates a number, n, and correctly predicts the value of the nth card from those dealt on the table.

2.3.2 Behind the Scenes

There are three components that allow this trick to work. First, spreading the deck on the table allows the magician to memorize one or more of the top five cards in the deck. Next, the deck is shuffled in a way that ensures the top cards remain intact [15]. This ensures that the memorized cards will be amongst the prediction candidates. Finally, the random number generator is engineered to "force" selection of one of the memorized cards.

To make this trick work, the generator has been built with two vulnerabilities. The first is an intermediate output that allows the magician to see what number would be generated, based on the current inputs. Knowing this allows the magician to cut off new inputs once a memorized card would be selected. Second, the generator uses a hash function and modulus to produce its output rather than a cryptographically secure method. This ensures that the magician's desired output will appear after a small number of inputs. Figure 1 shows the generator's use during the trick.

3 Study Instrument and Evaluation

Method The goal of our study was to judge how effective a non-traditional approach could be in teaching novices about our selected attacks. To do so, we prepared and presented a 90-minute Zoom session as an optional class for high school students in a remote-learning computer science summer camp. Using this particular format was a necessary workaround once COVID-19 restrictions

prevented the summer camp from being held live. We discuss the impact of this format switch on our study later in this section.

Once the true purpose of the trick is revealed, the presenter shared a brief lesson that named the attack, separated the attackers' real purpose from the misdirection stated while the trick was in progress (i.e. "creating a psychic bond with the deck"), and shared a real-world example. Though presented as one session for our study, each of the three modules could be the basis of a single classroom lesson.

To measure any change in the students' mastery of the material, we designed an assessment (a portion of which is shown in Table 1) consisting of 12 multiple choice questions (4 for each topic), 3 Likert-scale survey statements, and a free response section. The assessment, minus the Likert and free response questions, was conducted before the lesson to generate a baseline, and was repeated after the lesson to measure improvement and gather student opinions. In both cases, the participants completed the assessments online and outside of the workshop. We purposely avoided collecting demographic information on the respondents due to the heightened privacy concerns inherent in working with high school students.

	Question Text	Correct on Pre-test	Correct on Post-test
Q1	Which of the following is the best definition of social engineering?	3 (60%)	5 (100%)
Q2	The act of creating a scenario in order to extract information is called:	3 (60%)	4 (80%)
Q3	Which of the following pieces of information are dangerous to reveal online?	5 (100%)	5 (100%)
Q4	Bad actors can use stolen personal information to do which of the following:	2 (40%)	5 (100%)
Q5	What is a side channel attack?	0 (0%)	0 (0%)
Q6	Which of the following can give you a hint as to what a computer is doing?	5 (100%)	5 (100%)
Q7	What is an example of a common real-world side channel attack?	4 (80%)	5 (100%)
Q8	How could you prevent an attacker from stealing a password by using a microphone to listen to keystrokes?	3 (60%)	5 (100%)
Q9	Which of the following is a major use of hash functions?	4 (80%)	3 (60%)
Q10	Which of the following is an important feature of a good hash function?	4 (80%)	4 (80%)
Q11	When passing multiple items sequentially into a hash function, which item has the most influence on the output?	1 (20%)	5 (100%)
Q12	What is the term used when two or more inputs to a hash function generate the same output?	2 (40%)	5 (100%)

Table 1: Question text and aggregate scores for each assessment question. Q1-Q4 covered social engineering, Q5-Q8, side channel attacks, and Q9-Q12 attacks on randomness.

Results The limited number of assessments completed greatly limits the validity of our results, but does indicate positive trends. Aggregate scores increased across all categories on the post-test. The results in Table 1 show scores for the social engineering and attacks on randomness sections increased by 30%, while the side channel attacks section increased by 15%. The improvement in social engineering scores can be traced to higher scores on Q1, Q2, and Q4, indicating a better understanding of the topic. Smaller improvements on Q5 and Q9 suggest a need to improve the lesson materials in these specific areas, particularly providing better definitions and examples of side channel attacks real-world use cases for hash functions. On the plus side, accurate responses to Q11 and Q12 suggest the lesson was an effective scaffold for teaching two key properties of hash functions.

On the questionnaire, student shared very positive opinions about the lessons, attesting that the lesson had improved their skills in the covered topics, while also being enjoyable. Free response comments shared described the session as "fun," "entertaining," and "interesting." The instructor also observed that a significant majority of students kept their cameras on, and asked or answered questions about the material – two key indicators of engagement during remote instruction.

Limitations And Future Work COVID-19 restrictions, a remote modality, and difficulties handling consent forms drastically reduced participation from a potential enrollment of around 40 students to a group of 15 actual attendees. Of these attendees, only 10 agreed to participate in the study and just 5 completed it. The fall off in study completion can likely be attributed to an inability to do the assessment in person and to follow up about the post-test. It was simply too easy for students to sign off and forget to respond to the post test. This limited completion rate prevents us from making strong statistical claims about the effectiveness of our lessons. However, the positive responses observed by the instructor strongly suggest this approach could be successful in teaching cybersecurity topics.

4 Related Work

The idea of scaffolding is to provide a bridge to assist students in mastering material that may be beyond their reach [16] by bringing it into their "Zone of Proximal Development [12]." Given the complexity of computer science topics, it is not surprising that researchers have attempted to "scaffold" these concepts from a familiar base. In a meta-analysis from 2019, Szabo et al. identified 1283 papers in the field that contain scaffolding-related content [4], while Vanderyde et al. argues that increasing and more diverse enrollments in computer

science call for greater use of scaffolding practices [11]. Stanier also discusses using scaffolding approaches in higher education to support metacognitive and strategic skills [10]. All the above suggest our demonstrations could work as effective scaffolds for introducing security concepts to novices.

Other researchers have already integrated card tricks into computer science lesson plans, such as using parity bits to detect unintended bit flips, a central technique in error detection and correction. Bell et al. use a 5 by 5 grid of cards in an exercise that allows students to generate and detect parity errors. [1, 2]. Greenberg et al. were able to create more advanced versions of the exercise using larger grids. Other versions of this activity rely on software assistance to handle more complex computations [7].

In Ferreria et al. a "self-working" card trick called "Are You Psychic?" is used to explain topics in algorithm analysis and design, such as problem decomposition, pre- and post- conditions, and invariants [5]. Each of the trick's steps are mapped onto a formal description of an algorithm. Garcia et al. produced three papers describing a variety of magic tricks, along with the computer science concepts they help teach [6]. Their goal was to help students construct a mental model of how a computer actually works. Similarly, Curzon et al. found success explaining computer science concepts to younger students using magic shows [3].

5 Conclusion

In this paper we present a novel approach to teaching an introductory information security that uses card magic to simulate key attacks. By starting with a card trick, we are able to establish common ground even with students who have little knowledge of the field. The trick illustrates how the attack works giving the student a cognitive basis to build upon. After testing this lesson plan in a real-world teaching environment, we see its potential to foster engagement and improve students' mastery of the covered material. We encourage our fellow educators to use the tricks we have developed and to work out new ones as a way to make complex and intimidating material more approachable for novice students. Doing so could potentially improve not only individual performance, but also, by enhancing comprehension, reduce attrition rates among computer science undergraduates.

References

[1] Tim Bell, Jason Alexander, Isaac Freeman, and Mick Grimley. Computer science unplugged. *The New Zealand Journal of Applied Computing and Information Technology*, 13(1):20–29, 2009.

[2] CS Education Research Group. Cs unplugged: Error detection card flip magic. `https://classic.csunplugged.org/error-detection/\#Card_Flip_Magic`.

[3] Paul Curzon and Peter W McOwan. Engaging with computer science through magic shows. In *Proceedings of the 13th annual ITCSE conference*, pages 179–183, 2008.

[4] Claudia Szabo et al. Review and use of learning theories within computer science education research. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*, ITiCSE-WGR '19, page 89–109, New York, NY, USA, 2019. Association for Computing Machinery.

[5] João F Ferreira and Alexandra Mendes. The magic of algorithm design and analysis. In *Proceedings of the 2014 conference on Innovation & technology in computer science education*, pages 75–80, 2014.

[6] Daniel D Garcia and David Ginat. Demystifying computing with magic, part iii. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, pages 158–159, 2016.

[7] Ronald I. Greenberg and Dale F. Reed. Using magic in computing education and outreach. In *2018 IEEE Frontiers in Education Conference (FIE)*, pages 1–4, 2018.

[8] Our Pastimes. How to do the red and black separation card trick, 2017.

[9] Raymond and DeCourcey. *Cognitive Characteristics. Learners with Mild Disabilities.* Allyn & Bacon, A Pearson Education Company, 2000.

[10] Clare Stainer. Scaffolding in a higher education context. ICERI2015 Proceedings:7781–7790, 2015.

[11] James Vanderhyde and Florence Appel. With greater cs enrollments comes an even greater need for engaging teaching practices. *J. Comput. Sci. Coll.*, 32(1):38–45, October 2016.

[12] L.S. Vygotsky. *Mind in Society: The Development of Higher Psychological Processes.* Harvard University Press, London, 1978.

[13] Wikimedia Contributors. Svg playing cards, 2021.

[14] Wikipedia contributors. Forcing (magic) — Wikipedia, the free encyclopedia, 2021. [Online; accessed 12-August-2021].

[15] Wikipedia Contributors. Shuffling — Wikipedia, the free encyclopedia, 2021.

[16] David Wood, Jerome S Bruner, and Gail Ross. The role of tutoring in problem solving. *Journal of child psychology and psychiatry*, 17(2):89–100, 1976.

Computer Science Case Studies From the Census*

Christopher A. Healy
Department of Computer Science
Furman University
Greenville, SC 29613
chris.healy@furman.edu

Abstract

This paper describes some innovative assignments for CS 1 and CS 2 classes where students can write straightforward programs that discover useful facts directly from census data. This information exploits the geospatial population distribution of the United States. These assignments have been used successfully in Java and Python classes at this level, to reinforce skills in using file I/O and elementary data structures. In 2021, the U.S. Census Bureau began to release detailed results of the 2020 Census. This new data presents students with the opportunity to apply their programming skills to glean quantitative facts about the geographical distribution of the U.S. population and its diversity.

1 Introduction

Computing and the census share a long history. Every ten years the U.S. conducts a census of the population, and publishes extensive raw data. Herman Hollerith, whose firm was a corporate ancestor of IBM, developed a mechanical tabulator to read punch cards for the 1890 census. In 1946, the Census Bureau purchased the first commercially available electronic computer, the UNIVAC, for the 1950 census [2]. With the latest census having being conducted in 2020

*Copyright ©2022 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

and the results now being released, the time is ripe to consider incorporating the new data into computer science classes. This paper describes several software projects that have already been used in CS 1 and CS 2 classes based on data taken from the last census. By writing their own programs, students do not need to worry about the limitations of off-the-shelf software such as Excel or ArcMap.

2 The Data

The first data to be released by the Census Bureau is the data required by law to support redistricting [7]. For the purposes of disseminating data at various levels of detail, the United States is subdivided into the following census hierarchy: states, counties, census tracts, block groups, and blocks [1]. Summary statistics on these levels is given in Table 1. The averages and standard deviations have been rounded to the nearest integer. Note that only the census tract and block group have standard deviations that are significantly smaller than their means. This is because the Census Bureau makes a conscious effort to create these areas of relatively uniform size. Census tracts are commonly used by economists and sociologists to represent the intuitive idea of a neighborhood. On the other hand, the smallest geographical level in the census hierarchy is the block. The Census Bureau publishes only the most basic demographic data for blocks, such as race and sex, and how many people are aged 18 and older and thereby eligible to vote. For larger geographical areas, additional census data is available, such as income, educational attainment, and other economic statistics.

Table 1: Examples of small population areas in the US in 2020

Name of unit	Number	Mean population	S.D. of population
County	3,143	105,456	335,760
ZIP code (2010)	32,948	9,371	13,672
Census tract	83,848	3,953	1,689
Block group	238,437	1,390	682
Block	5,769,942	57	107

The Census Bureau also summarizes data by ZIP codes. These have the advantage that everyone knows their own ZIP code, while hardly anyone knows their census tract or block numbers. But there are some disadvantages to using ZIP codes for demographic analysis. ZIP codes were created for sorting mail, not for analyzing the population. ZIP codes exhibit a high degree of population variance, making them less suitable for our purposes. For example, many ZIP

code areas have populations of under 100 or over 100,000, making comparisons difficult. In addition, demographic data on ZIP codes from the 2020 census have not yet been released, because they are not needed for redistricting. As a result, Table 1 shows ZIP code data for 2010 instead of 2020. Fortunately, we do not need to resort to using ZIP codes, as there are online tools where one can quickly look up a census tract number, for example [8].

The Census Bureau's downloadable file format can be cumbersome to use directly by introductory students. The raw redistricting data from the 2020 census contained over 13 GB of data, spread across over 200 separate files. Therefore, it is recommended that the instructor perform some preprocessing of the raw census data, in order to create suitable input files for the students. For example, one input file listing all of the blocks, and another input file listing census tracts. For example, the modified block file created by the author has a size of 483 MB (uncompressed) and contains the following information:

- 5-digit state/county code to facilitate sorting
- 2-letter state abbreviation
- County name, truncated to 20 characters
- Tract number, which could be up to 6 digits omitting the decimal point
- Block number, 4 digits
- Population of the block, up to 5 digits
- White, Black, Hispanic, and Asian population of the block
- Latitude and longitutde, to the nearest thousandth of a degree
- Area of the block in acres

Fortunately, the format of the input data in 2020 data is the same as in 2010 [1]. Therefore, the procedure for scanning the 2020 data can be done seamlessly for 2010, making longitudinal comparisons straightforward.

The format of the modified tract file, having a size of just 7 MB, is analogous to the block file. Within both files, the fields have a fixed width, so that students can use a substring function in order to extract individual values, instead of having to tokenize.

3 Project Ideas

This section describes possible programming exercises, and most of these are suitable for CS 1.

3.1 Simple Analyses

A simple task to start with is finding a centroid or population center. It is the mean of the latitudes and of the longitudes, weighted by the population

of each area. To check their work, students are instructed to verify on a map that their answer is plausible. This task can easily be extended by computing the centroid of part of the U.S., such as a single state.

Next, using the block data, we can find the population density of each county or census tract. This way, we can classify areas as urban if their density exceeds 1000 people per square mile. While reading the data, the program needs to keep track of each tract or county encountered. So, students are recommended to use a built-in data structure such as the dictionary type in Python or the analogous Hashtable class in Java.

3.2 Create a Map

The map shown in Figure 1 was created with a program that simply reads the latitude-longitude locations of all census tracts from the 2020 census. A Java implementation is less than 70 lines long. The essential computation is to convert a latitude-longitude pair into a (x, y) pixel ordered pair. However, some care is needed to avoid creating a map that is upside-down or backwards. The lowest pixel numbers are in the upper-left corner of the image, while the lowest latitude and longitude values are in the opposite corner. As a sample classroom exercise, the author presented the source code to a CS 1 class where the output map was indeed displayed backwards, and the students were asked to find the error.

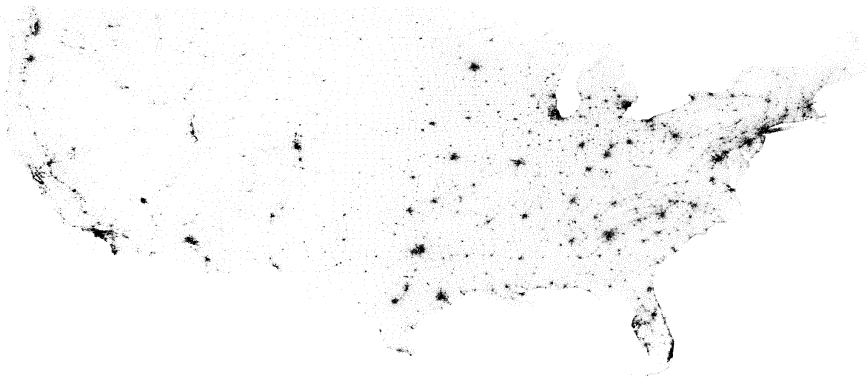


Figure 1: Pixel map of the US based on tract locations

Next, students are asked to modify this program, e.g. to change the resolution of the image, and to use blocks instead of tracts for additional detail. By reading the racial diversity counts of each block, a color-coded map can be created. Another use of color in a pixel map is to show areas of population

growth and decline. The program reads a second file containing the results of the previous census for comparison. The Census Bureau provides comprehensive block data for 1990, 2000, 2010, and 2020.

An alternative to a pixel map is to create a KML file that can be read by Google Earth. The Census Bureau publishes Shapefile data on all census tracts. Each tract is geometrically defined as a polygon with latitude-longitude vertices. However, due to the complexity of these shapes (i.e. the large number of vertices), it may be difficult to render more than one state's worth of census tracts in one map. A simpler map could be created by representing each tract abstractly as a simple shape such as a square.

3.3 Discover Population Clusters

Scanning the census data can answer many questions pertaining to the concentration of population. For example, The U.S. Department of Education publishes the latitude and longitude of every public school in the country. Since our census data also locates every block by latitude and longitude, we can find how many people live within a specified distance from each school. Croft et al. [4] similarly used census data to determine how many people live within five miles of a physician. On the flip side, we can also find areas of the country that are remote. Thus, we can identify populations that are underserved or to detect possible areas of low light pollution for astronomy.

3.3.1 Radius

In particular, the radius problem asks this question: Determine how many people live within a fixed radius, (e.g. 10 miles) of some point, such as a downtown area or real-time GPS coordinates. Note that this problem can be solved by a single pass of the input file, and that it is not necessary to sort the tracts or blocks by distance.

Because distance is a critical calculation in the radius problem, there needs to be an accurate way to estimate the distance between two latitude-longitude points. It is also necessary to convert from degrees to miles. Students are given a formula to use in their programs. It is a modified form of the Cartesian distance formula that assumes that the earth is a sphere.

Once the students understand how to compute the population of some circular region, the next step is to iterate this procedure for the entire country. For each census tract t, calculate the population $P(t, n)$ living within n miles of the center of t. In doing so, it is straightforward to find the population density within n miles of every census tract in the United States. By scanning the resulting output, it is then easy to find areas of high or low population density (i.e. urban versus rural). Many businesses prefer to locate themselves in areas

of relatively high population density. Therefore, the program could be run for a certain value of n, seeking a list of tracts where $P(t, n)$ is sufficiently high. For example, New York City has many tracts where a 10-mile radius encircles more than 7 million people.

3.3.2 Cluster

The cluster problem is analogous to the radius problem. The difference here is that instead of seeking a fixed distance from a certain point, we seek a fixed population size. For example, from a given point, how far do you need to go to encompass 50,000 people? This problem can be solved by sorting the tracts in ascending distance from the point in question. Since it is necessary to sort the areal units, it is important to use a smaller input file, such as census tracts, not blocks. Sorting all of the blocks in the United States would be overkill. If blocks are desired, then these should be limited to a single state or metropolitan area.

Once the clustering algorithm is implemented, then it too can be iterated over the entire country. The complete Python program used to perform the cluster analysis contains slightly over 100 lines of code. For each census tract t, it computes the radius $R(t, p)$ of the smallest circle centered at t containing a population a population of p. Then, the output can be scanned to find specific results of possible interest. To seek areas of high population density, one would search for a radius below some threshold. For example, if one wanted to find an area of 100,000 people in an urban density of at least 1000 per square mile, we need a circular area of area 100 square miles or less. Thus, we need a circle of radius 5.64 miles or less. Running the cluster program on the 2020 census data reveals that more than half of the census tracts in the country have this desired density.

One practical weakness of the radius and cluster programs is that they define only circular regions. As an alternative, a variation of the cluster problem is to partition the U.S. into nonoverlapping regions of similar population. This is similar to the real-world problems of redistricting and even the creation of census tracts themselves. In theory, the United States needs to be partitioned into 435 contiguous areas of equal population. In this case, there is also the added detail that each state is partitioned separately. Students can also focus on a single state alone to simulate the redistricting of a state legislature.

3.4 The Warehouse Problem

The warehouse problem is a generalization of finding a population center, and is presented to a CS 2 audience. The purpose of the problem is to find a set

of N centrally located points across the United States. The problem can be stated as follows:

"Company XYZ is in retail trade, and its management would like to build several warehouses around the country to store merchandise. The locations of these warehouses should be chosen so as to minimize the distances from these warehouses to the general public. For each potential customer in the United States, we wish to estimate the distance from that household to its nearest warehouse in order to minimize shipping costs. The output is a set of optimal locations for the warehouses."

Alternatively, rather than speaking of warehouses literally, the problem could be stated as seeking to minimize the distance that the public needs to travel to a location of business. The parameter to this problem is the desired number of locations. And of course, the scope of the problem can be limited to one state or metropolitan area rather than the whole country.

It is an optimization problem that selects an optimal sample of census tracts. Blocks are not used because the run time of the program would increase by a factor of 69 (the average number of blocks in a tract) for a gain in precision that we might not appreciate. Inside an urban area, a census tract is often about one square mile in size, which is sufficiently precise for this problem. Students are given a pseudocode algorithm that they implement in Java. The algorithm generates random samples of tracts, and among all the trials it finds the sample with the minimum average distance. It can be summarized as follows:

```
Create an array of Tract objects.
 Create array A of 10 Tracts for the optimal selection.
 For trial = 1 to 25,000:
    Randomly generate a selection S of 10 Tracts
    For each Tract t:
       d = shortest distance from t to any Tract in S
    Compute weighted average of all values of d
       (i.e. weighted by the population of t)
    If this weighted average distance < distance for the
       array A, then reset A to be the selection S.
       And keep track of its weighted average distance,
       and also the trial number where minimum was found.
    If trial is a multiple of 200, print out the current
       value of A and its average distance so that the user
       can observe the progress of the algorithm.
```

Students are given this pseudocode and asked to implement it in Java. Being able to implement pseudocode is a fundamental skill in computer science. Most students in the author's CS 2 class found this to be a nontrivial task. The two main stumbling blocks were understanding the concise language of

the pseudocode, and making sure that no detail was omitted. The run time complexity of the algorithm is linear in the desired number of warehouses and in the number of trials. The program is under 200 lines long. In practice, for 10 warehouses and 25,000 trials, students found the program to take about three minutes. In an upper-level algorithms course, students could explore other optimization strategies, such as a genetic algorithm.

As a further experiment, once students have written this program, they can run it on various numbers of business locations in order to discover a mathematical relationship between distance in miles, d, and the population size, p, served by each location. In other words, we can develop a rule of thumb for estimating the average distance that one needs to travel to the nearest establishment of some type. In our experiments, the regression formula we obtained was

$$p = 2481d^{1.8}$$

For example, if there is approximately one cardiologist for every 14,000 people in the U.S., we should expect the average American to live 2.6 miles from one.

4 Conclusion

The goal of this work was to make it straightforward for students in CS 1 and CS 2 to write programs that analyze the geographic distribution of the U.S. population and its diversity at a high degree of detail. The author has had success over the years at incorporating the 2010 census data into several different assignments and student research projects, and work has begun on doing the same for 2020. With the newly released 2020 census data, everyone now has the opportunity to begin analyzing the most recent demographic data. Furthermore, this work need not be limited to the U.S. Table 2 shows a short list of countries that also publish census data online along with corresponding geospatial data [3, 5, 6, 9]. Thus, all of the programs described in this paper could also be carried out on these countries as well.

Table 2: Examples of census hierarchies used in other countries

Country	Unit	Number	Mean size	S.D. of size
Australia	Statistical area 1	53,358	402	165
Australia	Statistical area 2	2,149	2,150	6,441
Australia	Statistical area 3	333	64,441	40,532
Brazil	Sector	310,120	615	354
Canada	Dissemination Area	54,963	627	536
Canada	Aggregate DA	4,920	7,003	3,965
New Zealand	Area unit	2,012	2,108	1,699

References

[1] United States Census Bureau. *2020 Census State Redistricting Data (Public Law 94-171) Summary File Technical Documentation*. 2021.

[2] Martin Campbell-Kelly and William Aspray. *Computer: A History of the Information Machine*. Westview Press, Boulder, Colorado, 2004.

[3] Statistics Canada. Census profile, 2016 census. `http://www12.statcan.gc.ca/census-recensement/2016/dp-pd/prof/details/page_Download-Telecharger.cfm`.

[4] Janet Croft, Hua Lu, Xingyou Zhang, and James Holt. Geographic accessibility of pulmonologists for adults with copd. *CHEST*, 150(3):544–553, 2016.

[5] Instituto Brasileiro de Geografia e Estatistica. Censo 2010 resultados. `http://censo2010.ibge.gov.br/resultados.html`.

[6] Australian Bureau of Statistics. Census advanced search. `http://www.abs.gov.au/websitedbs/censushome.nsf/home/map`.

[7] Congress of the United States. *Public Law 94-171*. 1975.

[8] Federal Financial Institutions Examination Council Geocoding System. `https://geomap.ffiec.gov/FFIECGeocMap/GeocodeMap1.aspx`.

[9] Statistics New Zealand. 2013 mesh block dataset. `http://www.stats.govt.nz/Census/2013-census/data-tables/meshblock-dataset.aspx`.

Porting the APTT MAGMA tool to an HPC environment using Singularity containers: A benchmark study*

Poster Session

E.K. Iskrenova-Ekiert[1], JT Haag[2], and Soumya S. Patnaik[3]
[1]Department of Computing Sciences
SUNY Brockport, Brockport, NY 14420
eiskrenovaekiert@brockport.edu
[2]HPC Intern, HPC Internship Program
DoD High Performance Computing Modernization Program
Lorton, VA 22079
[3]Aerospace Systems Directorate
US Air Force Research Laboratory
Wright-Patterson Air Force Base, OH 45433

This work was conducted during a summer project as part of the Department of Defense HPC Internship Program. The project was designed to introduce a Computer Science student to High Performance Computing and working in a multidisciplinary team of scientists and engineers. The project presented a challenging real-world problem frequently encountered in computational engineering involving a range of issues: porting a custom-built legacy code to a new computing environment, evaluating the code efficiency, interfacing different simulation packages, and performing large scale simulation runs in an HPC environment.

Model-based design is a powerful tool in physics-based system engineering that helps reduce design time and cost. Model-based design of increasingly complex aircraft thermal systems and architectures requires significant computational resources and necessitates the use of HPC systems. deploying custom-built Windows tools in an HPC environment can be challenging and container virtualization technologies provide a user-friendly solution that is portable.

Docker [7] and Singularity [6] are well-established container virtualization technologies that allow the researcers to package their custom application to-

*Copyright is held by the author/owner.

gether with all dependencies into a portable container that allows the custom application to run on a different machine and in a different environment without new compilation or any adjustments to the custom application. Containerization technologies provide new deployment strategies, ease of use, and the ability to execute applications in diverse computing environments.

Multipoint optimized Architecture Generator for Military Aircraft (MAGMA) [4, 2] is a MATLAB GUI-based tool, part of the Aircraft Power and Thermal Toolset (APTT) [5]. MAGMA has been developed to rapidly generate and analyze power and thermal architectures for aircraft conceptual design, to identify the best architectures given a set of system components and constraints. The goal of this project was to port the MAGMA tool to HPC environment and to carry out large-scale simulation runs that are otherwise not feasible nor practical on a consumer Windows desktop. We will discuss the challenges encountered in the course of this project-based learning experience, the approaches and solutions we designed, as well as, the lessons learned.

The first goal of the project was to create a Singularity container with an installation of OpenModelica [3] needed for MAGMA runs. Following the best practices for Singularity container development in HPC environment [1], a Docker container with OpenModelica v.1.16.5 within the Ubuntu 18.04 OS was created. On the HPC machine, we used Singularity to pull the container from Docker Hub. The porting of MAGMA to HPC environment involved adapting the MATLAB code to be executed at the command line i.e., avoiding the GUI parts of the code, as well as managing the library paths, since MATLAB prepends its libraries to the system library path. The third component of the project was to perform a large-scale production run by creating the appropriate PBS launch script using the OpenModelica Singularity container. This involved learning about computing environments on supercomputers and the PBS job scheduling.

We performed a timing study examining the efficiency of the APTT MAGMA code and determined the optimal set of parameters needed to perform large-scale runs. The best practices for performing MATLAB runs in an HPC environment suggest that the code should be compiled into a standalone app to eliminate the need for use of MATLAB and MATLAB Parallel Toolbox licenses. The code was modified accordingly and a standalone deployed APTT MAGMA app was compiled for use in an HPC environment with containerized OpenModelica. Additional code modifications were needed to allow for performing the runs in independent segments of user-defined size to help manage the size of the results files and to allow for asynchronous evaluation, thus further reducing the compute time. The standalone APTT MAGMA app was benchmarked by performing a production run analyzing 22056 hybrid thermal management system architectures in five segments. Each segment ran on 48

CPUs on the AFRL supercomputer Mustang and completed within five hours of walltime. The compiled MAGMA app was found to run 40% faster than within MATLAB. The study completed in less than 22 hours of walltime, a significant speedup compared to a month of runtime on a consumer Windows desktop.

In the course of this summer internship, we have successfully accomplished the major goals of the project: a Docker container and a Singularity container with OpenModelica v.1.16.5 have been produced and a large-scale APTT MAGMA study using the Singularity container was successfully carried out. A procedure for creating an OpenModelica Singularity container has been established and tested. The APTT MAGMA code was successfully ported to Linux and HPC environment and a standalone APTT MAGMA app which does not require the use of MATLAB licenses, was successfully deployed. A procedure for carrying out large-scale APTT MAGMA simulations has been created, tested, and documented. Additionally, this project significantly improved the efficiency of the APTT MAGMA tool and designed a procedure for compiling the standalone APTT MAGMA app that can be used for other MATLAB code in the future. *Approved for public release, Unlimited Distribution, Case Number: AFRL-2022-0137.*

References

[1] G. Behm. *PETTT Singularity Container Development Practices*. HPCMP PETTT/SAIC, Internal training presentation, 2019.

[2] R. Buettner, D.R. Herber, P.C. Abolmoali, and S.S. Patnaik. An automated design tool for generation and selection of optimal aircraft thermal management system architectures. In *AIAA Propulsion and Energy Forum (2021), Virtual Event*.

[3] P. Fritzson, P. Aronsson, H. Lundvall, K. Nyström, A. Pop, L. Saldamli, and D. Broman. The openmodelica modeling, simulation, and development environment. In *46th Conference on Simulation and Modelling of the Scandinavian Simulation Society (SIMS2005), Trondheim, Norway, October 13-14, 2005*, 2005.

[4] D.R. Herber, J.T. Allison, R. Buettner, P. Abolmoali, and S.S. Patnaik. Architecture generation and performance evaluation of aircraft thermal management systems through graph-based techniques. *AIAA 2020-0159*, 2020.

[5] E.K. Iskrenova-Ekiert, T.O. Deppen, D.J. Dierker, and S.S. Patnaik. Towards a common modeling environment for aircraft power and thermal systems design and optimization: Introducing the simulation platform aptt-sp. In *AIAA SciTech 2020 Forum (10.2514/6.2020-2118), January 6-10, 2020, Orlando, FL.*

[6] G.M. Kurtzer, V. Sochat, and M.W. Bauer. Singularity: Scientific containers for mobility of compute. *PLoS ONE*, 12(5):e0177459, 2017.

[7] D. Merkel. Docker: Lightweight linux containers for consistent development and deployment. *Linux J.*, 239(Article 2), 2014.

A Case Study for a Pilot Data Science Curriculum for Advanced High School Students*

Poster Session

Ching–yu (Austin) Huang[1], Janice Chao[2]
[1]School of Computer Science and Technology
Kean University, Union, NJ
chuang@kean.edu
[2]High Technology High School
Lincroft, NJ
janjan4fun@gmail.com

Data Science is quickly becoming one of the fastest growing and most critical areas in computer science. Extensive research and case studies have shown that it can significantly benefit businesses across all industries. In turn, the job market has a high demand for people with skills related to data analytics. Therefore, it is critical to develop a curriculum that can attract high school students to study computer science and data analytics and encourage them to pursue these fields in their career paths. The goal of this research project is to propose a non-programming curriculum that integrates the basic concepts of databases, statistics, and data mining for advanced high school students. This will allow students to experience the data science process - collecting, extracting, transforming and loading data to a normalized database, and perform statistical analysis to find the correlation between datasets that are stored in a database.

This study utilizes free software tools - MySQL database, MySQL Workbench IDE, and an online Chi-Square test calculator. Anxiety and skin disorder datasets were downloaded from the CDC website and converted from XPT to CSV format. Two tables were created in the database and datasets were uploaded to tables using MySQL Workbench. The student learned the basic SQL SELECT, WHERE and GROUP BY statements through the w3schools.com website to retrieve data from the database and generate the numbers for the

*Copyright is held by the author/owner.

2x2 table. A free online chi-square calculator was then used to produce the p-value, 0.038, a significant correlation between anxiety and skin disorder.

After 5 hours of basic instruction on SQL queries, MySQL Workbench, the Chi-square hypothesis, and the meaning of the significance, the high school student spent around 20 hours to self-study and practice these topics, in addition to 5 hours to find and understand the CDC datasets, and another 10 hours to search and review the citations. A total of 40 hours for this curriculum design should be allocated for high school students who already have an Algebra background. For the database component, the study suggests covering the basic SELECT, WHERE, AGGREGATE FUNCTION, GROUP BY, ORDER BY, INNER JOIN, and CREATE TABLE queries. It is not necessary to cover advanced topics such as functional dependency, normalization, and E-R diagrams for the high school curriculum. Overall, the results show this study is a very successful model that should be easily adopted by other advanced high school students.

User Experience and Visualization of Assistive Technology Devices*

Poster Session

Andres Arauz, Ching–yu Huang
School of Computer Science and Technology
Kean University
Union, NJ
{chuang,arauzgua}@kean.edu

As technology constantly emerges, keeping up–to–date with these changes can be a difficult task to accomplish. Technology has advanced on many fronts such as web development and user experience among others. The advancements in technology also include the rapid growth of assistive technology devices that can be utilized to increase independence in various aspects of a person's life, however with the amount of new assistive technologies it has become difficult for pediatric occupational therapy practitioners to keep up with them. We will focus in this paper on the use of new technologies for web development in order to present OT practitioners with a new tool to find new assistive technologies. Many technologies and frameworks that facilitate web programming have been developed by many technology companies such as Wix or Squarespace. These technologies which are vastly used nowadays provide users the availability to create websites, which facilitates the creation of new markets and informational sites to exist.

Thanks to the new technologies that facilitate the construction of informative websites, among other kinds. We decided to use Squarespace in our project thanks to its reliability and friendly user interface to investigate the usefulness these technologies present to those who are not tech-savvy while also digging deeper into new concepts that are involved in web development such as the benefits of an improved user interface, user experience (UX/UI) and search engine optimization (SEO).

In this project our goal was to explore these tools for web development and to understand more elaborate concepts of web development such as UI/UX while we research current assistive technology devices used by pediatric occupational therapy practitioners, we aim to give the opportunity for pediatric

*Copyright is held by the author/owner.

occupational therapy practitioners to maintain assistive device literacy in order to flourish in the robust world of technology. Additionally, we aim to reduce the time OT practitioners may spend when researching and selecting device options. By providing practitioners a resource that increases access to assistive device awareness and literacy, occupational therapy practitioners can further provide better health care quality and improve treatment outcomes for clients in need.

By the end of this project, we were able to create an informational website that is easy to find and use by OT practitioners where they are able to search by different categories among the many assistive technology devices compiled into the website. This created a new tool for practitioners to increase their knowledge about new and upcoming technologies that we hope will be vastly used in the future.

Discovering Ways to Increase Inclusivity for Dyslexic Students in Computing Education*

Poster Session

Felicia Hellems and Sajal Bhatia
School of Computer Science and Engineering
Jack Welch College of Business and Technology
Sacred Heart University
Fairfield, CT 06825
hellemsf@mail.sacredheart.edu, bhatias@sacredheart.edu

The years accompanying entrance into the university system are often characterized by a period of great transformation. These years can also be wrought with difficulties for many students, difficulties which are often compounded in students with disabilities (SWD). Reports from the U.S. Department of Education show that as recently as 2015-16, 19% of undergraduate students experienced some form of disability [1]. Additionally, statistics show that SWD tend to have lower post secondary completion rates than their counterparts [3]. A review of pertinent literature has shown that there still exist gaps within the field of computing education (CE) for teaching cybersecurity concepts to SWD. This poster is a continuation of the author's research into both the identification and analysis of the current educational methods in use within the field of CE for teaching concepts of cybersecurity to SWD. This poster aims at narrowing the scope of that research by performing a specific analysis of CE through the lens of the post secondary dyslexic SWD demographic.

This work began with a broad review consisting of an analysis of each chosen disability, followed by a focused literature review in the field of CE with emphasis placed on identifying current educational methods in use for teaching cybersecurity concepts to SWD. The criteria for the disabilities chosen for review was predicated upon the greatest impact on a students ability to both learn and perform tasks fundamental to cybersecurity. The following

*Copyright is held by the author/owner.
[1]https://nces.ed.gov/pubsearch/pubsinfo.asp?pubid=2021009

four disabilities were selected based on this criteria: Visually Impaired and Blind (VIB), Intellectual disabilities (ID), Autism, and Dyslexia. The initial review of literature identified two main patterns. The first identified pattern was related to how the materials were *integrated* into the curriculum, with methods falling largely into a short term or a long term model. Short term models were characterized by the delivery of methods in the form of camps or workshops that were short in temporal duration. Long term models delivered materials through amendments to current curriculum, creations of guidelines aimed at inclusivity, or adaptations to traditional teaching methods. The second identified pattern was related to how the materials were *implemented* into the curriculum, with prior work largely delivering these methods through the use of programming/coding or the use of tools. Emphasis in the discovered work related to teaching SWD concepts of cybersecurity has thus far largely been placed on VIB individuals with materials being delivered through short term integration models [2].

The review of literature revealed minimal work done within the field of CE in relation to methods for teaching dyslexic SWD concepts of cybersecurity. While there has been research performed in the fields of computing sciences relating to authentication and dyslexia, there has been little found in direct relation to CE for teaching SWD concepts of cybersecurity, indicating an existent gap in research focusing specifically on dyslexic SWD. Dyslexia can cause negative feelings for students surrounding academic self worth which may in turn negatively affect graduation rates [1]. Considering that traits of dyslexia may pose textual comprehension issues and difficulties decoding words, dyslexic SWD may shy away from cybersecurity studies that are largely textual in nature, such as cryptography [4]. These facts coupled with a need to increase inclusivity in CE underlines the necessity for further research regarding educational methods for teaching dyslexic SWD concepts of cybersecurity.

This noted gap in research spurs the authors goal for continued research aimed at increasing inclusivity for this demographic. Next steps include a study involving dyslexic SWD to gather statistical data regarding experiences in learning cybersecurity concepts. Having identified research performed in CE for teaching SWD involving the use of gamification through programming, the creation of a program focusing on a gamified assisted acquisition of cryptographical concepts that takes into account traits of dyslexia is underway. Integration of this gathered data will assist in producing an adequate reflection of the specific needs of dyslexic SWD in learning cybersecurity concepts. The adoption of this program in CE delivered through a long term integration model to ensure continuity of education will aid in not only increasing the presence of dyslexic SWD within the computing science field, but in maintaining higher rates of retention for this demographic in post secondary education.

References

[1] Maro Doikou-Avlidou. The educational, social and emotional experiences of students with dyslexia: The perspective of postsecondary education students. *International Journal of Special Education*, 30(1):132–145, 2015.

[2] Jesse R Hairston, Derrick W Smith, Tania Williams, William T Sabados, and Steven Forney. Teaching cybersecurity to students with visual impairments and blindness. *Journal of Science Education for Students with Disabilities*, 23(1), 2020.

[3] Lynn Newman, Mary Wagner, Anne-Marie Knokey, Camille Marder, Katherine Nagle, Debra Shaver, and Xin Wei. The Post-High School Outcomes of Young Adults with Disabilities up to 8 Years after High School: A Report from the National Longitudinal Transition Study-2 (NLTS2). *National Center for Special Education Research*, 2011.

[4] Linda S Siegel. Perspectives on Dyslexia. *Paediatrics & Child Health*, 11(9):581–587, 2006.

Teaching with VS Code DevContainers*

Conference Workshop

Stoney Jackson[1] *and Karl R. Wurst*[2]
[1]*Department of Computer Science and Information Technology*
Western New England University, Springfield, MA 01119
Stoney.Jackson@wne.edu
[2]*Computer Science Department*
Worcester State University, Worcester, MA 01602
Karl.Wurst@worcester.edu

A consistent development environment across students and faculty for a course, assignment, or a project is desirable to reduce unintended, time consuming, and frustrating failures that distract from intended learning goals. Faculty and institutions have tried various approaches to reduce these distractions (e.g., labs, laptop requirements, and remote access to servers). These problems are not isolated to academia. Industry also has also been struggling with these issues and has developed various practices and technologies to combat them. Most recently, container technology (e.g., Docker [2]) has emerged as a mechanism for packaging applications into relatively lightweight containers that isolate each application and its dependencies in a way that can be easily distributed to others. Visual Studio Code [4], with its Development Containers [1], has further expanded containers to packaging and distributing development environments customized for development of a specific application. Faculty have begun to leverage this technology to create and distribute to their students independent, lightweight, customized development environments for a particular course, example, or assignment [5, 6].

In this workshop, participants will learn how students would interact with development containers, and observe what it is like to download, install, and run a development container created by the facilitators. Participants and facilitators will have a candid discussion about the benefits and challenges of this technology. Participants will learn how to build development containers for their courses, and will build one with support from the workshop facilitators.

The facilitators have been developing open-source projects with their students, working with Docker for the past two years and have begun introducing Docker and development containers into their courses.

*Copyright is held by the author/owner.

Biographies

Stoney Jackson, PhD is a Professor in the Department of Computer Science and Information Technology at Western New England University, and is a maintainer for PLCC [3]. Karl R. Wurst, PhD is a Professor and Chair of the Computer Science Department at Worcester State University. They are founders and maintainers of LibreFoodPantry, a community of students and faculty building free and open-source software for food pantries. They are also members of the NSF-supported OpenPace project whose goal is to enrich computing education by engaging students in Humanitarian Free and Open Source Software (HFOSS). Recently, they have helped bring Docker and VS Code DevContainer technology to these projects and their Capstone, Database Applications, Programming Languages, and Software Development courses.

Acknowledgements

This material is based on work supported by the National Science Foundation under Grant Nos. DUE-2012990, DUE-2012999, DUE-1525039, DUE-1524877, and DUE-1524898. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF).

References

[1] Developing inside a container. `https://code.visualstudio.com/docs/remote/containers`. Accessed: 2021-12-09.

[2] Docker. `https://www.docker.com/`. Accessed: 2021-12-09.

[3] Programming language compiler compiler. `https://github.com/ourPLCC`. Accessed: 2021-12-09.

[4] Visual studio code. `https://code.visualstudio.com/`. Accessed: 2021-12-09.

[5] Pak Kwan. Let's dockerize our classrooms: Bringing the docker into the classroom: Pre-conference workshop. *J. Comput. Sci. Coll.*, 32(1):7–8, oct 2016.

[6] Sander Valstar, William G. Griswold, and Leo Porter. Using devcontainers to standardize student development environments: An experience report. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '20, page 377–383, New York, NY, USA, 2020. Association for Computing Machinery.

Initial Research: How Does Instructor Identity Change Due to Supporting Student Involvement in Open Source Computing for Good?*

Lightning Talk

Gregory W. Hislop[1], *Heidi J.C. Ellis*[2]
[1]*Computing and Informatics*
Drexel University
hislop@drexel.edu
[2]*CS & IT Department*
Western New England University
ellis@wne.edu

Instructor identity is the sense of self towards being an instructor that results from the beliefs and values that an individual holds. Instructor identity is shaped by how that self interacts with the affordances and limitations of the teaching and academic context and how that context impacts the sense of self. Instructor identity is important as it impacts job satisfaction, commitment to teaching, motivation and more.

One approach to exposing students to Computing for Good is for instructors to involve students in humanitarian free and open source projects (HFOSS). With guidance, students interact with open source communities to both learn about the community and, ideally, to contribute to the project. This use of HFOSS in education has shown potential to improve student motivation in studying computing. It also tends to impact instructional approaches including a push toward active learning, and a shift to mentoring and co-learning.

Instructor feedback indicates that this HFOSS education context has an impact on instructor identity. Some instructors report that they have changed their pedagogy as a result of supporting student involvement in humanitarian open source, with changes including taking on more risk in the classroom and allowing students to explore more freely [2].

*Copyright is held by the author/owner.

These reports of changes in pedagogy raise interesting questions related to the nature of identity [1] including:

- How do changes in pedagogy impact instructor identity?
- How does the culture of open source impact instructor identity?
- How do social interactions with open source communities impact instructors' identities?

This talk reports on the first steps towards exploring how instructor identity changes as instructors support student involvement in such projects. Initial observations will be described and a research plan will be outlined. Possible opportunities for instructor participation will also be presented.

References

[1] James Paul Gee. Chapter 3: Identity as an analytic lens for research in education. *Review of research in education*, 25(1):99–125, 2000.

[2] Lori Postner, Darci Burdge, Heidi JC Ellis, Stoney Jackson, and Gregory W Hislop. Impact of hfoss on education on instructors. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, pages 285–291, 2019.