Introducing Computing to a Cohort of Incarcerated Youth

Kirsten Mork, Theresa Migler, Zoë Wood
California Polytechnic State University
San Luis Obispo, CA
klmork,tmigler,zwood@calpoly.edu

ABSTRACT

Computer Science and programming are changing the world, but not everyone has equal access to education about this field. In California, Juvenile hall students typically lack opportunities to learn computer programming. In this paper, we present an experience report about the course we created to address the needs of this population. Specifically, we created and taught an introductory computing course focused on engagement. This project-based game design curriculum was launched to a small cohort of the juvenile hall in spring 2019. This course focused on engaging students while introducing computing programming concepts such as variables, logic and function. Student surveys and reports from their teacher showed this class had a positive impact and was well received by students and staff. We hypothesize and show initial positive indication that creative, game-oriented curriculum had a positive impact on the demographic. We also present some of the challenges encountered when working within the juvenile hall system and our solutions and general recommendations for these types of classes.

CCS CONCEPTS

• **Social and professional topics → Informal education**; **K-12 education**.

KEYWORDS

accessibility; gender and diversity; K-12 instruction; outreach; K-12 curriculum; incarcerated youth

ACM Reference Format:
Kirsten Mork, Theresa Migler, Zoë Wood. 2020. Introducing Computing to a Cohort of Incarcerated Youth. In *The 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20), March 11–14, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3328778.3366820

1 INTRODUCTION

There is a need for greater diversity and acceptance in the tech industry [3, 4] and providing access to CS education to a wide audience is an important part of the solution to promoting diversity in tech. Creating access to computing education requires broad initiatives at all levels of education. Even before college, stereotypes are embedded in K-12 students, limiting who will pursue a CS degree [5, 9, 10, 13]. In addition, only certain students in K-12 have access to CS curricula [6]. For example, in 2017 in California, only 3% of the state's 1.9 million high school students enrolled in a computer science course, with low income schools being four times less likely to offer AP computer science courses [2]. To achieve equity, there must be equal access to CS courses and students must be taught material tailored for all sorts of backgrounds, learning styles, and learning needs [11, 15].

This paper presents our experience designing and launching an introductory computer science course for the under-served population of incarcerated youth at a California Juvenile Hall. Given that the incarcerated youth population has very limited access to computer science education and that this education has benefits for all students, we believe it is important to introduce and study the effectiveness of this computing curriculum. We present information about our curriculum and initial evaluation of the course for this population. In particular, we wanted to validate whether the creativity of a project-based game design computer science curriculum was engaging and empowering for these students.

Research shows that young adults often believe the negative stereotypes associated with CS, such as CS being boring, only for the 'smart' students, antisocial, lacking creativity, and tedious [23]. A large part in creating equity is to break stereotypes before college; therefore, it is important to create more engaging experiences for all students. To effectively create curriculum for Juvenile Hall students, we relied on prior work teaching CS to elementary and high school students [7, 21], constructing engaging CS0 courses [16, 20], and creating accessible courses for non-majors [8]. In the end, we found our introductory course to be a positive experience for the students and staff. The students reported liking the class and expressed an interest in continuing to learn computer programming. They also reported liking the game design aspect of the course and indicated they wished the course was longer.

2 SETTING

We partnered with an organization that educates the community on and practices restorative justice. One of the ways they provide restorative justice is through offering a wide variety of classes by bringing in individuals with a broad range of skills to teach at the local Juvenile Hall and Jail.

The work described in this paper builds on prior work with them which focused on teaching Python to incarcerated adults in the area [14]. In order to teach in these environments, the curriculum must meet certain guidelines. For example, the student computers can not have access to WiFi during the class and computers are only available to the students *during* the class. All practice and homework assignments must be done by hand, allowing learning to continue outside the classroom, but still meeting the specific needs of the situation.

As this prior course was designed for adults, for this project we developed curriculum specifically for the younger audience, found in Juvenile Hall, but with similar constrains and goals as the prior work.

Demographics: Within Juvenile Hall, there were multiple subsets of students. We chose to work in a setting that supported longer incarceration (6-12 months) for middle or high school students where the students would be there long enough to teach recurring classes to. These students are "moderate to high risk and in need of residential treatment". Therefore, a curricular focus was on making sure the class was empowering and fun. The curriculum was designed to challenge, but include immediate encouragement and reward as well.

However, there were also many volatile variables for the class including the size of the class, the exact grade of the students and the uncertainty of when students would join or leave, thus, we incorporated flexibility into the curriculum to allow for adaptation to the needs and interests of the students as they arose; we also always had extra activities planned in case we needed to change course. Finally, we had well documented base code, worksheets, and activities prepared for students who joined late in the course. The curriculum assumes that students have basic typing skills, but could be adapted as needed.

3 THE CURRICULUM

Research has found that project-based learning is engaging for non-majors and can increase self-efficacy [8, 12]. In addition, there is a need for both breadth and depth in an introductory CS curriculum [16, 18]. Finally, research shows the benefits of proximal subgoals and immediate rewards, as proximal subgoals (close, attainable subgoals) over distal goals (larger, further away goals) caused intrinsic interest and personal efficacy [1]. We attempted to incorporate these findings through a 2D game design course.

Research on teaching high school students shows many young adults have stereotypes that CS is anti-social and boring; we utilized the researchers' suggestions for correcting those misconceptions by using game elements, fun projects, real-world applications, and creativity [23].

3.1 Curricular Design

Overall, the curriculum was built around creating a simple 2D game. This way, the students would have a project to work on which they could hopefully be proud of and feel ownership of, as recommended by researchers mentioned above. Throughout the five weeks, each class consisted of a 10-20 minute lesson to teach a new coding concept and then an assignment or two to practice the new concept. Each assignment would directly add to the students' games. This way, they would practice the lessons of that day while still working towards an immediate subgoal for their own project.

We chose to use Processing, a Java-based language for 2D graphics. Other researchers have found success in using Processing for introductory courses due to its simplicity and immediate visual feedback [7, 19–22]. In addition, it is a text-based language, which avoids some transitional issues some students experience later on when learning in a block-based language [17].

One goal, based on research mentioned above, was to incorporate creativity and game design into our curriculum. The students were allowed to design their game hero given the constraints of the commands they knew and design or choose all of the elements in their game. In addition, although we had instructions and outlines for their game, as much as possible we tried to accommodate the student requests to customize different elements of play.

Course Goal: Overall, the goal of this curriculum is to give opportunity to provide an introduction to computer science to an under-served population, specifically that of juvenile hall students. As an introductory course and an independent elective for high school students, the goal of our curriculum is retention and inclusion over technical depth.

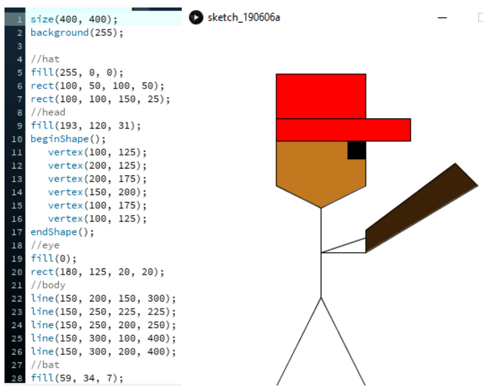
Structure: Coding is a difficult skill to learn, and often requires multiple facets to practice. While typical CS0 classes might have lectures, lab assignments, homework, and projects, this class was limited to two one-hour periods each week for five weeks. In two hours a week, we needed to provide all of the lectures and assignments, projects, and work periods. The students were not able to work on the course material outside of the hour blocks, partly due to their schedules and partly because we provided the laptops used by the students each week. Bringing in laptops that could not connect to the internet made the process much smoother in the Juvenile Hall environment. Also, when originally planning the course we did not know if there would be other computers available to the students. Without being able to assign homework, it was essential to have every assignment work towards the overall game the students were making. We hypothesize this was more engaging, working towards a goal instead of doing isolated practice problems; it was necessary to make time for all of the lectures, labs, and project work in an hour period. We discuss some techniques used to manage the timing constraints below.

3.2 Lessons

Overall, there were ten classes over the course of five weeks. The course outline is included below.

Lesson 1 - Introduction to Computer Science: This lesson introduced students to the goals of the class as a 2D game design course and how it fit into the context of CS as a whole. It also examined the various fields of computer science as well as a very high-level overview of software versus hardware, code, programming, and programming languages. Specific to the goals of this class we discussed 2D space, and interactively graphed points in a 'normal' grid versus screen-space. Students were asked to come up to the board to plot points in the two different spaces. Finally, we outlined the 2D game they would make, a simple game where a hero slides across the screen to collect falling objects. *Assignment 1:* Students were asked to design the main hero for their game, with constraints on number and type of shapes. The students were given a grid to draw on and to label their shapes.

Lesson 2 - Processing Commands to Draw Hero Following the design of the hero on paper, this class focused on the Processing commands to enable the students to program the drawing of the hero using code. The presentation included screen size, rgb colors and shape commands, relating each of these to painting and visual art creation. Even this simple introductory topic allows for a demonstration and discussion about command ordering. For example, you must pick a color before painting, and in programming the developer must write the color command before the draw shape commands. Similarly, for layering shapes to create a compound hero from basic shapes, the order the shape commands are issued effects the drawing order. Students were given a reference worksheet with commonly used commands to assist them while they coded since their laptops did not connect to the internet. *Assignment 2:* Students were asked to write code to draw their individually designed hero. A student example can be seen in Figure 1.



```
size(400, 400);
background(255);

//hat
fill(255, 0, 0);
rect(100, 50, 100, 50);
rect(100, 100, 150, 25);
//head
fill(193, 120, 31);
beginShape();
   vertex(100, 125);
   vertex(200, 125);
   vertex(200, 175);
   vertex(150, 200);
   vertex(100, 175);
   vertex(100, 125);
endShape();
//eye
fill(0);
rect(180, 125, 20, 20);
//body
line(150, 200, 150, 300);
line(150, 250, 225, 225);
line(150, 250, 200, 250);
line(150, 300, 100, 400);
line(150, 300, 200, 400);
//bat
fill(59, 34, 7);
```

Figure 1: Example of Student Work: Drawing a Hero in Processing.

Lesson 3 - Functions: This lesson focused on functions, explaining the built in *setup* and *draw* functions, necessary for 2D animation with Processing, as well as individually created functions written for other purposes. *Prep Work:* To help move the class along in total, all but the last few commands of the 'hero' code the students began in lesson two were completed by the instructor prior to class. In addition, base code was provided to enable 2D animation (a *setup* function and a *draw* function, where the 'hero' code was placed). By helping prepare this individual base code for students, we were able to efficiently introduce functions at the beginning of lesson three, as well as to start teaching the students how to read code that was not written by them. In addition, we also wrote a separate example program with three different functions to draw three different background scenes. *Assignment 3:* Students finished their heroes, becoming familiar with the new code organization and reminding themselves of the Processing commands. Next, students were instructed to apply what they learned about functions to call the associated function to draw the background they desired in their game. This allowed students to practice using functions, reading code, and integrating preexisting code into their program, meanwhile saving them time and still allowing them some creative choice.

Lesson 4 - Continuing Functions and Introduction to Variables In this lesson, we reviewed functions and students were taught coding concepts to allow them to move their hero to the bottom of the screen. Students were introduced to the topic of variables with a white-board lesson. This was followed by a demonstration in Processing, which shows how a variable representing position can be changed to make shapes move across the screen. *Assignment 4:* Students were asked to move their hero code from the *draw* function to a new *hero* function. They also were asked to translate the hero to the bottom left of the screen. In addition, students designed the falling objects for their games. These objects would ultimately be collected by their heroes in a later lesson.

Lesson 5 - Continuing Variables This lesson began with a white-board introduction to variable scope. Parameters, local variables, and global variables were all discussed along with the difference between integers and floats. *Assignment 5:* The students were asked to use their knowledge about variables to enable the position of their hero to change, resulting in it moving across the bottom of the screen.

Lesson 6 - Conditionals and Booleans: This lesson introduced conditionals and booleans and their use in Processing. *Assignment 6:* The students were asked to use conditionals to make their hero stop moving when they reached the right side of the screen. Next, they used booleans to make the hero move back and forth along the bottom of the screen.

Lesson 7 - IO Key Controls and More Fun with Variables Students were introduced to input and output especially related to key board and mouse interactions typically used in 2D games. *Prep Work:* Again to help move the class along, the instructor wrote the code for the students' 'falling objects' based on the designs they made in lesson four. In addition partial base code for recognizing key presses was provided. *Assignment 7:* The students were instructed to complete the code associated with key presses to allow them to control their hero's movement. They also were given an assignment to review variables, since the topic was still confusing for some of them. They practiced by creating different position variables for each instance of a falling object. Next, they used these variables to draw and animate as many falling objects as they wanted on their screen at a time.

Lesson 8 - Game Play Class Overview: In contrast to the original game structure with the hero collecting falling objects, some students indicated interest in creating a dodging game instead, where the hero had to avoid falling objects and would lose if they were hit by one. To accommodate the student requests, we made lesson eight an outline of game play options. We gave an overview on how to make a game-over screen when an object hit the ground. We also gave an overview on how to make the falling objects cycle back to the top of the screen when they hit the ground. *Assignment 8:* Students were given individual assignments based on the game variants they chose. This assignment required the use of booleans, basic draw functions, and using conditionals to check when objects reached the edge of the screen. These were all topics students had previously learned.

Lesson 9 - Collision Detection: Class Overview: The lesson focused on how to represent collisions in a game using bounding box approximations. This was demonstrated with pseudo-code and drawings on the whiteboard. *Assignment 9:* Students began implementing collision detection.

Lesson 10 - Final Touches Class Overview: This lesson was designed for students to focus on the fun, final touches of game creation. Students were given a free work period. They were also given the option to demo their game to the class. *Prep Work:* Again due to time constraints, individual assistance was provided to complete the collision detection from the prior lesson. *Assignment 10:* Students added final edits to their games.

3.3 Challenges and Considerations

This educational experience for students had a strict time limit due to students not being able to access the computer lab outside of class time. To balance learning new concepts and game development, we needed to be aggressive about time management. One way we supported students with the class time constraints was by having two other teaching assistants (Cal Poly students) attend each lesson to assist, providing a one-to-two ratio of TAs to students during most of the course. We found the class would have been very difficult without TAs being able to provide individual help. While the students stayed engaged during the lecture and learned overall concepts during that time, 10-20 minutes limited the amount of technical learning. Therefore, the tutoring-style work period allowed students to get the individual help needed to make progress on their projects and aided in reiterating the lessons.

Another way we accommodated for the time constraint was providing scaffolding for different parts of the project. In addition, while each student designed their project and coded the first part of each assignment every class, one hour was never long enough to learn and then finish an assignment on a new topic. With the students' permission, as noted above some prep work included us finishing the remainder of some of the class assignments and adding new base code to prepare for the next lesson. This allowed students to be ready to move on to the next lesson. The decision to provide individualized assistance to move along through the curriculum this way emerged after the first few classes and after talking with the students' regular teacher. He explained it would be most encouraging for the students if, once they started learning a new concept, they were not bogged down by tedious tasks when time was so limited. From his experience working with this student population, he emphasized that holding their engagement was of utmost importance. We admit this is a limitation to student learning as tedious work can be an important part of learning; however, we feel this was the correct solution when balancing engagement and learning (as shown by student responses to post-survey questions).

Implementing this course required some specific and important details that we share here for those considering implementing a similar program. We needed a good deal of lead time prior to starting the class due to the sensitive nature of the population. Volunteers needed to be cleared to enter Juvenile Hall and trained on rules and regulations. Another setup requirement was determining how to make this course count for the students' high school credits, as each student had different course requirements remaining. Some students were able to count this course as a general elective credit; however others needed it to be a math elective or art course instead. The range of course needs of these students supports the idea of incorporating multiple subjects while teaching CS curriculum. Since our course was a combination of 2D game design and computer science, it was heavy in math, coding, and art/design; this enabled it to count for diverse course needs.

4 VALIDATION

To assess the impact of this experience for students, surveys were given both before and after the class to gather student efficacy and opinions. The surveys were anonymous and were given in the form of printed out Google form surveys (for the pre-survey) and questions on the white board (for the post-survey). This research received IRB approval with care given to working with incarcerated youth.

The number of students in the course varied over the course offering due to the nature of Juvenile Hall and students varying attendance there. Overall, seven different students participated in the course. The sample size for this result is too small for these findings to be statistically significant, however, we found their responses informative, especially when considering our future plans for the next iteration of this course.

4.1 Quantitative Results

The following were the main questions our surveys addressed:

Research Question 1: Is our proposed intervention, specifically using game design, an effective and engaging way to introduce the target demographic to CS?

Our hypothesis when designing this curriculum was that game design would be the most engaging way to teach this group. However, our pre-survey results showed that students had the exact same interest in learning to program as they did in learning to make games. This implies that other areas may have been equally interesting to these students.

However, although pre-surveys indicate games are no more enticing than computer programming in general, post-surveys show that most students would recommend a game focus over computational art for future iterations of this class. Four of the six students present for the post-survey suggested game design, and the other two reported no preference. (Computational art was shown to be an engaging context for a different high school population [21], thus it was under consideration for future iterations).

When students were asked to rank on a Likert scale, with 1 meaning 'disagree' and 7 'fully agree', how much they liked the class, the average response was 7. We note, however, that the survey was hand-written, thus although the average score was 7, not all students responded with 7. One student reported a 6 while another student reported liking the class 8 out of 7.

Research Question 2: What preconceived opinions do students in this demographic have concerning CS, games, coding, and creativity?

In the pre-survey, students reported on whether they would consider a career in computer science and if they thought they would be able to get a job in computer science one day if they wanted to. A seven-point Likert scale was used for students' responses (1 being 'disagree' and 7 being 'fully agree'). Students mostly had neutral or positive responses to these statements. Specifically for considering a career in CS, all responses were >= 3, with an average of 4.75. For the question about being able to get a CS job, all responses were >= 2, with an average of 4.5. While we only have data for 4 students for the pre-survey, more students joined the class after the first day.

Students showed a more positive outlook when asked if they would like to learn computer programming and if they would like to learn how to make games on the computer, with an average response of 5.5 for both questions.

Finally, students were asked if they thought they were creative and if they liked when their classes let them be creative. Students, once again, had fairly positive responses with the average response of 5.75 for both questions related to creativity (all responses were >= 4). This further supports the idea of using creativity to engage a broader range of students in CS.

Research Question 3: Does student interest in CS increase after completing the course?

Although we did give pre and post surveys, only three students of the seven total were present for both the pre-survey and post-survey. For each of the three students, they showed very positive responses in the post-survey, with each student improving their perspective on both considering a career in computing and continuing with learning computer programming. The results for the three students, when asked on a 7-point Likert scale if they would consider a career in CS and like to learn (or continue to learn) computer programming, can be seen in table 1.

	Pre-Course Surveys	**Post-Course Surveys**
	Consider CS Career	**Consider CS Career**
student 1	6	7
student 2	4	7
student 3	6	7
	want to learn CS	**want to learn CS**
student 1	7	7
student 2	6	7
student 3	4	7

Table 1: Comparison of Pre and Post Surveys for the Students Who Took Both (Other Students Joined Late or Left Early) (Seven-Point Likert Scale)

4.2 Qualitative Results

When exploring the research questions qualitatively, we found positive results as well. Overall, everyone involved - the students' usual teacher, the students, and the volunteers - reported the class as being a positive experience. The teacher was excited to have his students interacting with college students and hopes to continue this course in the future, as well as have other students from other majors teach.

4.2.1 Positive Student Engagement. The students all seemed to enjoy the class, as we had positive experiences with every student despite their initial levels of interest in the subject. For example, one student showed very little interest at first and wondered why he had to take the class as he already had chosen a career path outside of computer science; however, by the end of the course, although he still did not want to pursue computer science as a career, he was one of the most engaged students, working until the very end of class on the last day. He took initiative with his work, researching new commands and coming up with out-of-the-box ideas we had not taught. He even learned how to add lives and a high score counter on the last day, although this was outside of the scope of the class lectures. Overall, even though he does not wish to pursue this career path, he reported he would like to continue learning computer programming.

In addition, all students showed signs of having pride and excitement in their work. The last day, we gave students the option to demo their game to the class. While the students declined to demo in front of the class, every student, during the last class period and also throughout the previous classes, demonstrated their games to each other in one-on-one settings. The students would regularly and excitedly call the teacher, the volunteers, the guards, and each other over to their tables to show off new elements they had added to their games.

4.2.2 Benefits from Course Flexibility and Tutoring Style Work Periods. We found flexible curriculum and tutoring-centered work periods helpful in this class setting. Flexible game goals and having many opportunities for one-on-one help allowed the students to focus on the areas of the game they were most interested in, while less interested students still had basic, functioning games by the end of the course. One student, for example, was most interested in the artistic and design aspects of making a game. Although he did not pick up complex coding concepts as fast as some students, he worked very hard and stayed focused on his work. He was the only student who asked if he could change the default backgrounds given to him and spent much of his time perfecting it. A screen-shot of his game is included in Figure 2.

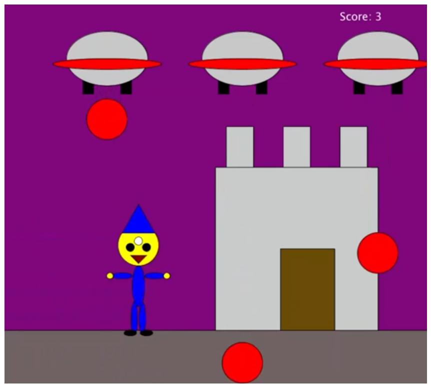


Figure 2: Screenshot of Student's Game.

In addition, we still found the class to be a positive experience for students joining late. In Juvenile Hall, it is very common for attendance to fluctuate. Students who joined the course halfway through were given base code and able to integrate into the class. They were given extra help from tutors and other students and were able to complete a game by the end. However, one student showed up so near the end, we were unable to teach him how to make a functioning game. For this student, we taught him the basic draw commands and had him make static drawings with Processing. By the end of the class, he reported that, although it seemed difficult, he liked programming and would want to continue to learn.

4.3 Threats to Validity

At this time, we are presenting our experience with introducing a computer programming course to incarcerated youth. Overall, the experience was a highly positive one. We present information gained in our pre and post surveys, but acknowledge that the sample size of this data is very small. There were only a total of seven students who participated in the course throughout the lessons. This means our survey results are not strong forms of evidence nor are they statistically significant. Furthermore, only *three* of the students took both the pre and post surveys for comparison. However, the focus of this study was not quantitative results. The value was in pioneering a new course and making the connections to make it sustainable. Future work could include a heavier focus on quantitative analysis once this course is more established.

Another potential threat to validity was unexpectedly meeting the students prior to the first day of class. Though it was helpful to meet them and introduce the course topic prior to the first lesson, this was an unplanned event and potentially inflated student responses on the pre-survey; by the time we gave them the pre-survey asking their opinions on computer science, they had already been informed of some benefits of computer science to prep them for the first day of class. However, although skewing the data, meeting the students early was beneficial and made the first class easier.

Finally, although we instructed the students to keep the pre and post surveys anonymous, some students wrote their name on the post-survey. Since they chose to disclose their name, they may not have answered as honestly.

5 CONCLUSION AND FUTURE WORKS

In summary, we have presented our experience report on creating and teaching an introductory CS course for incarcerated youth. This course attempted to empower students that previously did not have access to any computing courses through a creative, project-based curriculum. We designed our class around 2D game design, with an emphasis on creativity, course flexibility, and tutoring-style work sessions. We found the course to have initial success, as all students reported enjoying the class and a desire to continue to learn computer programming. The students also reported liking the game design aspect of the course and stated that they wished the course was longer.

For anyone considering a similar course and likewise for our future courses at Juvenile Hall, it is essential to plan early, plan for flexibility, and have lots of volunteers. We found it was very beneficial to incorporate multiple subjects to adapt to different students' course requirement needs. We also found an effective class setup to be a short lecture followed by a short assignment. Students expressed that they wished the class was longer than five weeks, so future iterations we plan to expand the course to allow students more time to work with computing concepts and project creation.

Although this initial course offering was successful, we plan a more vigorous study over time to experimentally validate our initial findings. For example, while we intentionally incorporated creativity into the curriculum, we were not able to specifically investigate the effects of varying degrees of creativity, i.e. we cannot assert how important the creative components of our curriculum were.

ACKNOWLEDGMENTS

To Restorative Partners, whose work has changed the lives of many and made our work possible as well. To the local teacher and staff at Juvenile hall who were also essential for this program. To Timothy Wong and Erik Mork for being wonderful tutors.

REFERENCES

[1] Albert Bandura and Dale H. Schunk. 1981. Cultivating competence, self-efficacy, and intrinsic interest through proximal self-motivation. *Journal of Personality and Social Psychology* 41, 3 (1981), 586 – 598. http://search.ebscohost.com.ezproxy.lib.calpoly.edu/login.aspx?direct=true&db=pdh&AN=1982-07527-001&site=ehost-live

[2] Kapor Center and the Computer Science for California (CSforCA) coalition. [n.d.]. Computer Science In California's Schools: An Analysis of Access, Enrollment, and Equity. Accessed: 2019-15-08.

[3] LaVar J. Charleston, Phillis L. George, Jerlando F. L. Jackson, Jonathan Berhanu, and Mauriell H. Amechi. 2014. Navigating underrepresented STEM spaces: Experiences of Black women in US computing science higher education programs who actualize success. *Journal of Diversity in Higher Education* 7, 3 (2014), 166 – 176. http://search.ebscohost.com.ezproxy.lib.calpoly.edu/login.aspx?direct=true&db=pdh&AN=2014-37844-002&site=ehost-live

[4] Sapna Cheryan, Victoria C. Plaut, Paul G. Davies, and Claude M. Steele. 2009. Ambient belonging: How stereotypical cues impact gender participation in computer science. *Journal of Personality and Social Psychology* 97, 6 (2009), 1045 – 1060. http://search.ebscohost.com.ezproxy.lib.calpoly.edu/login.aspx?direct=true&db=pdh&AN=2009-22579-003&site=ehost-live

[5] Sapna Cheryan, Sianna A. Ziegler, Amanda K. Montoya, and Lily Jiang. 2017. Why are some STEM fields more gender balanced than others?. *Psychological Bulletin* 143, 1 (2017), 1 – 35. http://search.ebscohost.com.ezproxy.lib.calpoly.edu/login.aspx?direct=true&db=pdh&AN=2016-48466-001&site=ehost-live

[6] K-12 Computer Science Framework Steering Committee. 2016. *K-12 Computer Science Framework*. Technical Report. New York, NY, USA.

[7] Katie M. Davis, Zoë Wood, and John Wilcox. 2016. Eighteen Hours of Code with Fifth Grade Students (Abstract Only). In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)*. ACM, New York, NY, USA, 694–694. https://doi.org/10.1145/2839509.2850554

[8] Jessica Q. Dawson, Meghan Allen, Alice Campbell, and Anasazi Valair. 2018. Designing an Introductory Programming Course to Improve Non-Majors' Experiences. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18)*. ACM, New York, NY, USA, 26–31. https://doi.org/10.1145/3159450.3159548

[9] Alexandria K. Hansen, Hilary A. Dwyer, Ashley Iveland, Mia Talesfore, Lacy Wright, Danielle B. Harlow, and Diana Franklin. 2017. Assessing Children's Understanding of the Work of Computer Scientists: The Draw-a-Computer-Scientist Test. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17)*. ACM, New York, NY, USA, 279–284. https://doi.org/10.1145/3017680.3017769

[10] Alexandria K. Hansen, Ashley Iveland, Cameron Carlin, Danielle B. Harlow, and Diana Franklin. 2016. User-Centered Design in Block-Based Programming: Developmental & Pedagogical Considerations for Children. In *Proceedings of the The 15th International Conference on Interaction Design and Children (IDC '16)*. ACM, New York, NY, USA, 147–156. https://doi.org/10.1145/2930674.2930699

[11] Päivi Kinnunen and Lauri Malmi. 2008. CS Minors in a CS1 Course. In *Proceedings of the Fourth International Workshop on Computing Education Research (ICER '08)*. ACM, New York, NY, USA, 79–90. https://doi.org/10.1145/1404520.1404529

[12] Ahmad M. Mahasneh and Ahmed F. Alwan. 2018. The Effect of Project-Based Learning on Student Teacher Self-Efficacy and Achievement. *International Journal of Instruction* 11, 3 (2018), 511 – 524. http://search.ebscohost.com.ezproxy.lib.calpoly.edu/login.aspx?direct=true&db=eric&AN=EJ1183424&site=ehost-live

[13] Allison Master, Sapna Cheryan, and Andrew N. Meltzoff. 2016. Computing whether she belongs: Stereotypes undermine girlsâĂŹ interest and sense of belonging in computer science. *Journal of Educational Psychology* 108, 3 (2016), 424 – 437. http://search.ebscohost.com.ezproxy.lib.calpoly.edu/login.aspx?direct=true&db=pdh&AN=2015-37516-001&site=ehost-live

[14] Theresa Anne Migler-VonDollen and Lizabeth T Schlemer. 2018. Engagement in Practice: Teaching Introductory Computer Programming at County Jails. In *2018 ASEE Annual Conference & Exposition*. ASEE Conferences, Salt Lake City, Utah.

[15] Laurie T. O'Brien, Donna M. Garcia, Alison Blodorn, Glenn Adams, Elliott Hammer, and Claire Gravelin. 2019. An educational intervention to improve women's academic STEM outcomes: Divergent effects on well-represented vs underrepresented minority women. *Cultural Diversity and Ethnic Minority Psychology* (2019). http://search.ebscohost.com.ezproxy.lib.calpoly.edu/login.aspx?direct=true&db=pdh&AN=2019-22448-001&site=ehost-live

[16] David Reed. 2001. Rethinking CS0 with JavaScript. *SIGCSE Bull.* 33, 1 (Feb. 2001), 100–104. https://doi.org/10.1145/366413.364552

[17] William Robinson. 2016. From Scratch to Patch: Easing the Blocks-Text Transition. In *Proceedings of the 11th Workshop in Primary and Secondary Computing Education (WiPSCE '16)*. ACM, New York, NY, USA, 96–99. https://doi.org/10.1145/2978249.2978265

[18] Elizabeth Schofield, Michael Erlinger, and Zachary Dodds. 2014. MyCS: CS for Middle-years Students and Their Teachers. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE '14)*. ACM, New York, NY, USA, 337–342. https://doi.org/10.1145/2538862.2538901

[19] Zoë Wood. [n.d.]. Fifth grade Introduction to Computer programming using Processing - PCS. http://users.csc.calpoly.edu/~zwood/Outreach/PCS.html. Accessed: 2019-02-08.

[20] Zoë J. Wood, John Clements, Zachary Peterson, David S. Janzen, Hugh Smith, Michael Haungs, Julie Workman, John Bellardo, and Bruce DeBruhl. 2018. Mixed Approaches to CS0: Exploring Topic and Pedagogy Variance after Six Years of CS0. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education, SIGCSE 2018, Baltimore, MD, USA, February 21-24, 2018*. 20–25. https://doi.org/10.1145/3159450.3159592

[21] Zoë J. Wood, Paul Muhl, and Katelyn Hicks. 2016. Computational Art: Introducing High School Students to Computing via Art. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education, Memphis, TN, USA, March 02 - 05, 2016*. 261–266. https://doi.org/10.1145/2839509.2844614

[22] Dianna Xu, Aaron Cadle, Darby Thompson, Ursula Wolz, Ira Greenberg, and Deepak Kumar. 2016. Creative Computation in High School. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)*. ACM, New York, NY, USA, 273–278. https://doi.org/10.1145/2839509.2844611

[23] Sarita Yardi and Amy Bruckman. 2007. What is Computing?: Bridging the Gap Between Teenagers' Perceptions and Graduate Students' Experiences. In *Proceedings of the Third International Workshop on Computing Education Research (ICER '07)*. ACM, New York, NY, USA, 39–50. https://doi.org/10.1145/1288580.1288586