Experiences Teaching Coral Before C++ in CS1

Frank Vahid
Computer Science and Engineering
University of California, Riverside
Riverside, California, USA
vahid@cs.ucr.edu
Also with zyBooks

Kelly Downey
Computer Science and Engineering
University of California, Riverside
Riverside, California, USA
kldowney@ucr.edu

Lizbeth Areizaga
Computer Science and Engineering
University of California, Riverside
Riverside, California, USA
larei002@ucr.edu

Ashley Pang
Computer Science and Engineering
University of California, Riverside
Riverside, California, USA
apang024@ucr.edu

ABSTRACT

Coral was introduced several years ago to ease the learning in college-level introductory programming courses. Coral consists of a simple textual code language and corresponding flowchart language and a free web-based educational simulator. Previous researchers described the benefits of Coral in CS0 courses and the first weeks of CS1 courses. We previously used Coral in CS1 and enjoyed the teaching experience, due to: the simple intuitive syntax, the simulator's auto-creation of a flowchart from code, and the simulator's visualization of code and flowchart program execution. However, we wanted to ensure we weren't hurting students with the transition from Coral to C++. This paper describes our experiences of teaching Coral in a ~100-student CS1 section for weeks 1-3 versus two other sections that taught C++ only. We performed analyses to answer three research questions: (1) Do students learn Coral more easily than C++? (2) Do students easily transition from Coral to C++? and (3) Do Coral-treated students do equally well on later C++ programs? We analyzed performance on auto-graded code-writing problems in zyBooks. We did not find support for (1), but did find support for (2) and (3), with Coral-treated students easily switching to C++ and performing equally well on later C++ programs. We conclude that CS1 instructors who enjoy the early-weeks teaching benefits of Coral can do so confidently knowing that students will perform equally well later in the course.

CCS CONCEPTS

• Social and professional topics - Professional topics - Computing education - Computing education programs - Computer science education - CS1

KEYWORDS

CS1, Programming, Coral, Simulator, Flowcharts, Pseudocode

SIGCSE 2023, March 15–18, 2023, Toronto, ON, Canada.

ACM ISBN 978-1-4503-9431-4/23/03. https://doi.org/10.1145/3545945.3569732

ACM Reference format:

Frank Vahid, Kelly Downey, Lizbeth Areizaga, and Ashley Pang. 2023. Impact of Several Low-Effort Cheating-Reduction Methods in a CS1 Class. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2023), March 15--18, 2023, Toronto, ON, Canada. ACM, New York, NY, USA,* 6 pages. https://doi.org/10.1145/3545945.3569732

1 INTRODUCTION

Coral is a language designed in 2017 via collaboration among two universities and a company to fill a gap in intro college-level programming courses [1]. The gap was between syntax-free block-based graphical languages like Scratch, Snap, and Alice popular among K-12 learners [2, 3, 4] and syntax-focused commercial textual languages like Python, Java, and C++ used in college-level intro programming courses. Block languages are perceived by some students as beneath college-level [5], and some students have trouble transitioning from blocks to textual languages [6, 7]. On the other hand, Python, Java, and C++ were made for professionals and have syntax or rules that are hard on beginners. Ex: In Python3, x = input(), then print(x + x), with input 2, outputs 22, not 4.

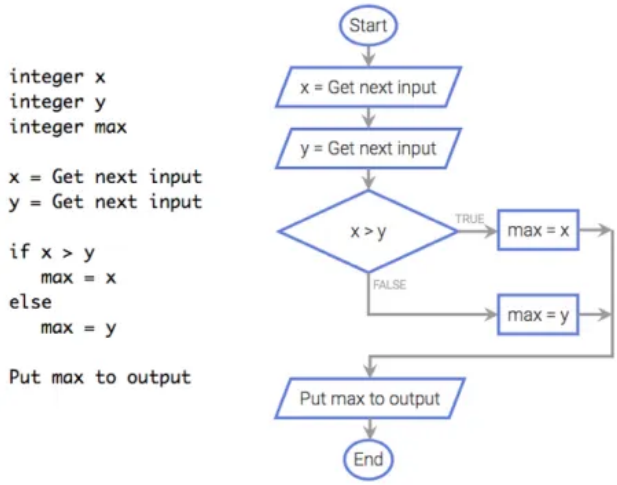

Figure 1: A Coral example: Outputting the max of two input numbers. (Image source: CoralLanguage.org).

Coral's code version has an executable pseudocode-like syntax with just 9 constructs for variables, input, output, assignment, branches, while loops, for loops, functions, arrays and with just a couple data types. Each construct has an equivalent flowchart syntax. Figure 1 shows a simple example using Coral's code and flowchart languages. Figure 2 shows the free web-based simulator [8], which shows variable values, input/output, and step-by-step program execution on either the code view (shown) or flowchart view (which is auto-derived from the code). Coral was used by 20,000+ students in 2021 [8].

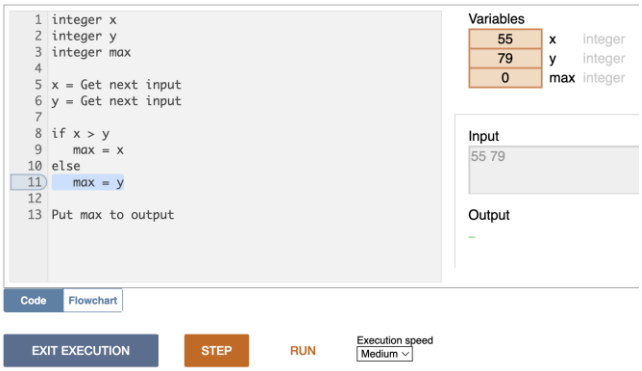

Figure 2: Coral's web-based simulator.

Coral enforces rules that aim to reduce non-logic-focused decisions for learners, such as all indents being 3 spaces, one statement per line, and all declarations before statements. Coral is strongly typed. Coral's simulator gives learner-focused feedback for syntax errors, such as:

> "Integer" is not recognized. Did you mean: integer? Note: Capitalization matters.

McKinney [9] found better grades using Coral in CS0 vs non-executable pseudocode or flowcharts in Raptor [10].

Previous research on CS1 courses has examined using learner-focused languages like Scratch, Snap, or Alice in early weeks before transitioning to a commercial textual language like Python, Java, or C++, but problems exist [7]. For example, Powers [11] found students were confused due to different object models and frustrated having to deal with syntax errors in the textual language, and performed less well after transitioning vs. a comparison group. Garlick [5] found students were frustrated having to learn a language that wasn't a "real language". In contrast, using Coral in CS1 prior to C++, Allen [12, 13] found students liked the language and simulator and performed equally well on an identical final exam as C++-only students. But, they taught Coral for 5 weeks and found some students would have preferred to start C++ sooner.

In this work, we taught Coral for our CS1's first 3 weeks, then switched to our main language of C++, and had an excellent teaching experience. We make frequent use of the simulator during lectures and office hours to help students visualize sequential execution, storage and updating of variable values, and branch and loop execution in both code and flowchart views. Students indicate appreciation for the simulator too. But, we wanted to know if students were learning more easily and if the transition to C++ was going smoothly, neither of which was addressed in previous work. Furthermore, we wanted to know how Coral-treated students did on later C++ programming tasks vs. C++-only students (and not just doing well on the final exam as in previous work). This paper provides analyses aiming to answer those questions.

2 CS1 AND CORAL USE

2.1 CS1

Our CS1 is at a 30,000-student public state "R1" (research active) university, being a mature course, teaching about 1,500 students per year, half computing majors and half non-majors (mostly required to take CS1 by their science/engineering major). The 10-week quarter course teaches C++ with weekly topics (before we started using Coral) generally being: I/O, Assignments, Branches, Loops(1), Loops(2) + Strings, Midterm, Functions(1), Functions(2), Vectors, File I/O + Classes, Classes + Misc.

The course uses zyBooks [14] for reading, homework, and programs, configured so that every week is one chapter. Every week follows the same pattern: "reading" with ~100 learning questions (Participation Activities or PAs, due before Tuesday's lecture), ~20 homework problems (Challenge Activities or CAs, either code reading or code writing to complete a small program, due Friday night), and 5-8 programming assignments (Lab Activities or LAs, typically with solutions 20-50 lines each, due Sunday night). PAs, CAs, and LAs are all in the zyBook, and are auto-graded with immediate feedback, partial credit, and unlimited resubmissions (until instructor-set deadlines if any).

2.2 Coral use

zyBooks has similar intro programming content for both Coral and C++ (among other languages). We configured our zyBook to combine Coral and C++ content. Our initial attempt three years ago involved 4.5 weeks of Coral: I/O + Assignments, Branches, Loops + Arrays, Functions(1), Function(2). The end of Week 5 had a Coral-only midterm, and the remaining 5 weeks taught C++, redoing all the above topics plus strings and a few additional topics. While overall a good experience, many students were eager to start with C++ sooner, and some struggled with the C++ programs compressed into 5 weeks. Thus, we now teach 3 weeks of Coral before switching to C++, as shown in Figure 3. In Week 4, the topics in Weeks 1-3 are covered again but in C++.

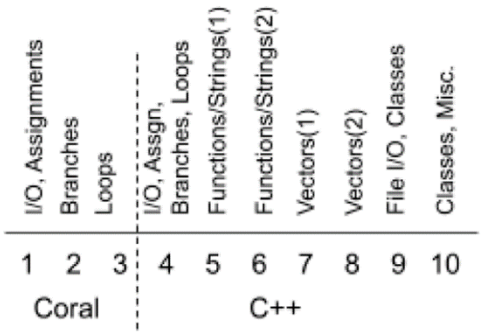

Figure 3: Our CS1 now teaches Coral in Weeks 1-3 up to loops, then switches to C++ in Week 4.

This Coral approach was used Fall 2021 in a ~100-student section and compared with two ~100-student C++-only sections that quarter, to address our research questions.

3 ANALYSES

3.1 Do students learn programming more easily in Coral than in C++?

We enjoyed the first weeks' teaching experience using Coral, largely due to the easy syntax, the visual step-by-step simulator, and the auto-creation of flowcharts. However, we wished to also test the following hypothesis:

- H1: In the first weeks of learning programming, students spend less time learning basic programming concepts using Coral than using C++.

To compare, we focused on a particular kind of zyBooks Challenge Activity known as a "progression CA", whose features include: (1) multiple parts of increasing challenge, and (2) each part's problem is auto-generated. We focused on progression CAs because the auto-generation greatly reduces the confounding that may occur on other programming tasks where students might be copying from classmates or from online solutions websites. zyBook progression CAs either involve code reading ("What does this code output?") or code writing ("Complete this code to do X)"; we focused only on code writing progression CAs.

We found four Coral CAs in our 3-week Coral content nearly identical to C++ CAs in the early weeks of the C++-only sections. More specifically, we found several Coral CA parts that were nearly identical to C++ CA parts. Results are shown in Figure 4, comparing time spent by students, which we determined using CA log data provided by zyBooks. The logs have time stamps for every run. We computed the time spent using the difference between timestamps, ignoring breaks of 10 minutes or more.

CA	C++ score	Coral score	C++ time	Coral time	C++ runs	Coral runs
Writing output	1.00	1.00	8.2	2.5	11.2	7.8
Calling math functions	0.99	0.98	1.8	2.6	4.2	7.8
Writing if branches	0.98	0.98	5.9	6.3	9.4	23.9
Writing if-else branches	0.94	0.97	7.1	6.9	8.7	13.7
Average	0.98	0.98	**5.7**	**4.6**	8.4	13.3

Figure 4: Coral students vs. C++ students on nearly-identical progression CA parts, in the first few weeks of the quarter. Time is in minutes.

Based on time spent, the data does *not* support the hypothesis. Writing output seemed easier in Coral, but the activities with more logic seemed about the same. Coral students did not spend *more* time either. These results match research comparing block-based and textual languages for learners where a research meta-analysis showed insignificant differences [15]. It seems that the difficulty of learning the logic of programming overshadows the difficulty of learning commercial language syntax.

The data did yield an interesting point: Coral students ran code more than C++ students for two CAs having branches. This is likely due to students using the simulator to visualize step-by-step execution of the code and flowchart views. In contrast, the C++ CAs simply show the code's results (the student presses "Run", causing compilation/running on a cloud server, with the output results being returned). Even with those additional runs by Coral students, the total time solving those coding problems did not increase.

3.2 Do students easily transition?

A concern is the Coral to C++ switch may cause students trouble as they mix up syntax. Our Coral section switched to C++ in Week 4, in which Coral-treated students did many of the same content sections that the C++-only students had done or would be doing. Many of those C++ content sections were review for Coral students, nearly identical to Coral sections but using C++ instead. Our hypothesis was:

- H2: Coral students would not take more time doing C++ CAs during the transition in Week 4, and would achieve the same scores, vs. the C++-only students doing those same CAs.

We examined that week's C++ CAs and found several that also appeared in the C++ section's zyBook. Figure 5 provides results.

CA	C++ score	Coral score	C++ time	Coral time	C++ runs	Coral runs
Writing output	1	0.99	8.2	3.4	11.2	5.8
Calling math functions	0.99	1.00	1.8	1.2	4.2	3.4
If branches	0.98	0.97	5.9	4.2	9.4	6.2
If-else branches	0.94	0.98	7.1	5.7	8.7	6.2
Average	**0.98**	**0.99**	**5.7**	**3.6**	8.4	5.4

Figure 5: Coral-treated students doing C++ CAs in Week 4, versus C++-only students doing those same CAs. Time is in minutes. Coral-treated students do not spend more time as was the concern.

The data supports the hypothesis. Coral students did not spend more time, and in fact spent less time (3.6 min vs. 5.7 min on average, or 40% less) due to those CAs being a review of concepts with different syntax. Coral students achieved virtually the same score (0.99 vs. 0.98). The data suggests Coral-treated students transitioned easily.

3.3 Do Coral students do equally well on later C++ programs?

We wanted to ensure the early Coral treatment did not harm students' learning of C++. We had the following hypothesis:

- H3: Coral-treated students will perform equally well on later C++ programs as C++-only students, achieving similar scores in similar times.

CA	C++ score	Coral score	C++ time	Coral time	C++ runs	Coral runs
For loops	0.99	1.00	4.3	5.0	6.5	7.1
Functions with loops	0.95	0.89	9.6	14.8	10.6	14.3
Check password	0.88	0.93	9.4	7.8	9.4	9.2
String manipul.	0.92	0.99	7.0	5.7	8.2	8.0
Vectors	0.75	0.70	18.7	16.8	14.3	14.3
Average	**0.90**	**0.90**	**9.8**	**10.0**	9.8	10.6

Figure 6: Coral-treated students doing C++ CAs in latter weeks, versus C++-only students doing those same CAs. Time is in minutes. Coral-treated students do not perform worse as was the concern.

Allen [12] previously compared Coral-treated students with C++-only students on final exam performance and found no difference, thus supporting the hypothesis. Here, we examine performance on progression CAs. Our data also supports the hypothesis. Coral-treated and C++ students both achieved the same scores (averaging 0.90 out of 1.0) and spent nearly identical time (10.0 min vs. 9.8 min).

4 THREATS TO VALIDITY

The Coral section was taught by a different instructor (Instructor A) than the two C++ sections (Instructor B). The instructor differences could have impacted the analyses. For H1, perhaps Coral students would have learned more easily but Instructor A's weak teaching negated any benefit. For H2 and H3, perhaps Coral students would have struggled but A's great teaching compensated. But, Instructors A and B are both experienced (over 10 CS1 terms each) with strong teaching evaluations and consistent grades. Beyond that, both instructors taught Spring 2022 CS1 as one course, using the same zyBook, syllabus, exams, graders, etc., and *both* used the early-Coral approach. Figure 7 shows results on the same CAs. Students performed similarly across Instructors A and B, which increased confidence that the different instructors were not strongly confounding (p-value for time was 0.55, and runs 0.72, far from 0.05 for statistical significance, using a two-tailed unpaired t-test).

CA	Score (A)	Score (B)	Time (A)	Time (B)	Runs (A)	Runs (B)
1	0.99	1.00	3.5	5.7	14.8	22.9
2	0.99	0.98	2.1	2.7	7.8	7.9
3	0.97	0.98	5.1	6.7	20.5	23.4
4	0.98	0.98	7.2	6.4	14.1	11.9
5	1.00	0.98	2.9	3.6	5.8	6.0
6	0.99	0.99	1.5	1.3	4.1	3.8
7	0.97	0.99	4.3	3.1	7.3	5.3
8	0.98	1.00	5.5	5.7	6.0	6.4
9	1.00	1.00	2.3	2.8	4.1	4.3
10	0.96	0.99	4.0	4.2	5.9	6.5
11	0.91	0.99	10.7	12.6	12.0	13.3
12	0.95	0.89	7.4	8.8	7.8	6.7
13	0.95	0.86	8.5	11.4	6.5	8.8
Average	**1.0**	**1.0**	**5.0**	**5.8**	**9.0**	**9.8**

Figure 7: Results on the same CAs as earlier, but in a later quarter when Instructors A and B taught the same class. No significant difference is observed.

Ideally, in the Fall 2021 comparison, the Coral-treated and C++-only students would have taken the same C++ final exam on which students could then have been compared. However, as the course was transitioning from the Covid-pandemic era back to regular classes, Instructor A chose to continue with online programming exams (auto-graded), while Instructor B switched back to the regular in-person written exams (half multiple choice, half code writing with manual grading). Due to the different exam modalities, though the two groups both did about the same on

their exams, drawing conclusions from that comparison did not seem appropriate, so we do not report those results.

This analysis was performed on progression CAs rather than LAs, because we have found progression CAs are less likely to involve cheating due to generating unique problems for each student. Ideally, we would also analyze the larger "Lab Activity" (LA) programs. But, for LAs, great care must be taken to control cheating because students in CS1 classes (nationwide) are known to copy from classmates, to obtain solutions from online sites, to hire contract programers, etc. Because Coral is not as widely used as C++, it might be expected that C++ students would have more ability to copy solutions or hire contractors, potentially skewing results. Furthermore, for the terms under consideration, our different CS1 sections involved differences in the prevention, detection, and punishment of cheating on LAs, also skewing results. However, analysis of LAs would be an interesting future work.

6 TIME SPENT

zyBooks provides instructors with per-student time data for PAs, CAs, and LAs. As additional analysis beyond this paper's main focus, Figure 8 shows that time data for the Coral-treated and C++-only students in the weeks preceding the midterm (Weeks 1-5). As a reminder:

- Coral students studied: 1 Coral I/O/Assignments, 2 Coral Branches, 3 Coral Loops, 4 C++ I/O/Assignments, Branches, Loop, 5 C++ Functions/Strings.
- C++-only students studied: 1 I/O, 2 Assignments, 3 Branches, 4 Strings/Loops, 5 Loops.

The time data indicates that both groups of students spent roughly the same time in Weeks 1-3, with Coral students spending slightly more. However, the data shows that the Coral-treated students spent more time during Week 4 (the transition week), but then C++ students spent a bit more in Week 5. We plan to investigate ways to smooth Week 4's time commitment for Coral-treated students; the number of CAs in particular might be a good target for reduction, and we might reduce LAs as well, perhaps combining some.

7 CONCLUSIONS

As in dozens of university courses, in our CS1 course, we have begun teaching Coral in the first weeks then transitioning to a commercial language (in our case, C++). We found the teaching and learning experience using Coral to be excellent largely due to Coral's simple learner-focused code syntax, the auto-creation of flowcharts from the code, and the free online education-focused Coral simulator. We found that Coral students don't spend significantly less time doing their auto-graded coding homework problems (CAs) in early weeks -- perhaps there is simply a minimum time needed to learn programming logic, and C++'s more complex syntax doesn't impose too much of a barrier in the

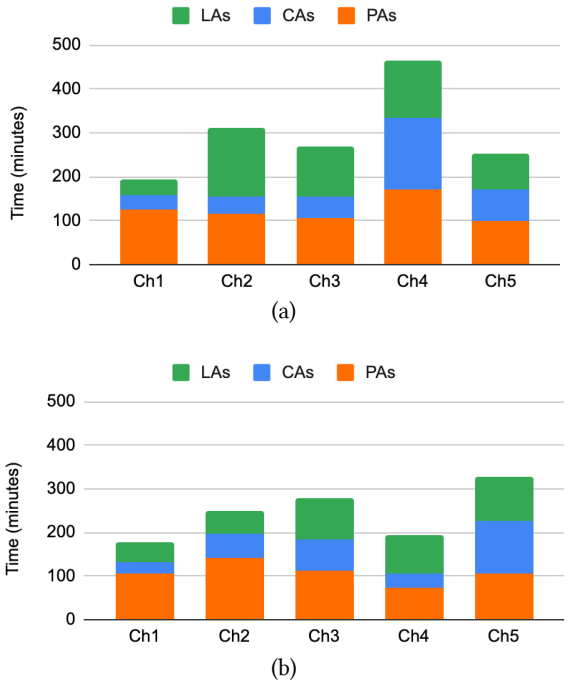

Figure 8: Weekly time spent prior to the midterm: (a) students using Coral in Weeks 1-3, then transitioning in Week 4 by redoing content in C++, (b) C++-only students.

early weeks. Coral students did conduct more runs, without spending more time, for the CAs involving branches, suggesting they were making good use of the educational simulator. We found Coral students easily transitioned to C++, spending no more time doing the C++ CAs during the transition week -- in fact, spending 40% less time on the particular CAs being compared, as those CAs were largely a review for them. Doing such a review is a strategy some professors follow intentionally, along a spiral learning process. We also found that Coral students did equally well on later C++ CAs, suggesting no harm in their learning of C++ imposed by learning Coral first. As such, instructors wishing to experiment with using Coral in the first weeks of their CS1 before teaching a commercial language, perhaps to ease students nerves, to make use of Coral's free educational simulator, and/or to level the playing field a bit regarding prior programming experience (since most students won't already know Coral), can likely do so confident that their students will transition easily to the commercial language and will learn the commercial language equally well.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 2111323.

REFERENCES

[1] Edgcomb, A.D., Vahid, F. and Lysecky, R., 2019, June. Coral: An ultra-simple language for learning to program. In 2019 ASEE Annual Conference & Exposition.

[2] Cooper, S., Dann, W. and Pausch, R., 2000. Alice: a 3-D tool for introductory programming concepts. Journal of computing sciences in colleges, 15(5), pp.107-116.

[3] Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. and Kafai, Y., 2009. Scratch: programming for all. Communications of the ACM, 52(11), pp.60-67

[4] Harvey, B., Garcia, D.D., Barnes, T., Titterton, N., Armendariz, D., Segars, L., Lemon, E., Morris, S. and Paley, J., 2013, March. Snap!(build your own blocks). In Proceedings of the 44th ACM technical symposium on Computer science education (pp. 759-759).

[5] Garlick, R. and Cankaya, E.C., 2010, June. Using Alice in CS1: A quantitative experiment. In Proceedings of the fifteenth annual conference on Innovation and technology in computer science education (pp. 165-168)

[6] Moors, L., Luxton-Reilly, A. and Denny, P., 2018, April. Transitioning from block-based to text-based programming languages. In 2018 International Conference on Learning and Teaching in Computing and Engineering (LaTICE) (pp. 57-64). IEEE.

[7] Blanchard, J., Gardner-McCune, C. and Anthony, L., 2020, February. Dual-modality instruction and learning: A case study in CS1. In Proceedings of the 51st ACM Technical Symposium on Computer Science Education (pp. 818-824).

[8] CoralLanguage.org, accessed 2022.

[9] McKinney, D., Edgcomb, A.D., Lysecky, R. and Vahid, F., 2020, June. Improving pass rates by switching from a passive to an active learning textbook in cs0. In 2020 ASEE Virtual Annual Conference Content Access.

[10] Carlisle, M.C., Wilson, T.A., Humphries, J.W. and Hadfield, S.M., 2004. Raptor: introducing programming to non-majors with flowcharts. Journal of Computing Sciences in Colleges, 19(4), pp.52-60.

[11] Powers, K., Ecott, S. and Hirshfield, L.M., 2007, March. Through the looking glass: teaching CS0 with Alice. In Proceedings of the 38th SIGCSE technical symposium on Computer science education (pp. 213-217).

[12] Allen, J.M. and Vahid, F., 2020, June. Teaching Coral before C++ in a CS1 Course. In 2020 ASEE Virtual Annual Conference Content Access.

[13] Vahid, F., Allen, J.M., Edgcomb, A.D. and Lysecky, R., 2020, July. Using the free Coral language and simulator to simplify first-year programming courses. In 2020 First-Year Engineering Experience.

[14] zyBooks.com, accessed 2022.

[15] Xu, Z., Ritzhaupt, A.D., Tian, F. and Umapathy, K., 2019. Block-based versus text-based programming environments on novice student learning outcomes: A meta-analysis study. Computer Science Education, 29(2-3), pp.177-204.