Keep Calm and Code on Your Phone: A Pilot of SuaCode, an Online Smartphone-Based Coding Course

George Boateng
ETH Zürich
gboateng@ethz.ch

Victor Wumbor-Apin Kumbol
Nsesa Foundation
vkumbol@nsesafoundation.org

Prince Steven Annor
New York University
prince.annor@nyu.edu

ABSTRACT

Africa lags behind the rest of the world in terms of digital literacy skills with less than one percent of African children leaving school with basic coding skills. One cause of this gap is poor access to equipment such as computers for teaching and learning. Yet, there is a proliferation of smartphones in Africa. Seeking to leverage this opportunity, we developed SuaCode, an online smartphone-based coding course to teach programming fundamentals to Africans. We designed the course to teach coding in a visual, interactive and fun way through the building of a pong game using Processing (a Java-based programming language). In this work, we describe our experience delivering the course online to 30 Ghanaian high school and college students. At the end of the course, 7 of the 30 students completed the first part of the course, building the pong game. The reflection essays from our students showed that they enjoyed the course and coding on a smartphone was not a barrier to completing the assignments. Improvements such as having more mentors and automated feedback on the coding assignments will improve the quality of the course. Given the difficulty in accessing computers in Africa, our work shows that smartphones can be leveraged to effectively introduce students to programming concepts via an online course. We are excited about the results of this pilot and see the potential to scale the course to eventually bring coding skills within arm's reach of millions across Africa, literally into their palms thereby bridging Africa's digital divide.

CCS CONCEPTS

• **Applied computing → Distance learning**; **E-learning**; Learning management systems.

KEYWORDS

smartphones, mobile phones, online course, coding, introductory programming, Processing, Ghana, Africa

ACM Reference Format:
George Boateng, Victor Wumbor-Apin Kumbol, and Prince Steven Annor. 2019. Keep Calm and Code on Your Phone: A Pilot of SuaCode, an Online Smartphone-Based Coding Course. In *Proceedings CSERC 2019 18-20 November 2019 Computer Science Education Research Conference Larnaca, Cyprus (CSERC '19), November 18–20, 2019, Larnaca, Cyprus.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3375258.3375260

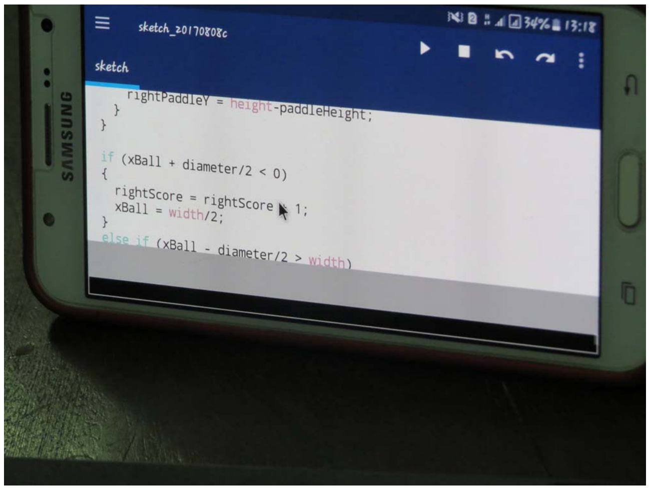
1 INTRODUCTION

Africa lags behind the rest of the world in terms of digital literacy skills, with less than one percent of children leaving school with basic coding skills [13]. One cause of this gap is poor access to equipment such as computers for teaching and learning, which hampers efforts to promote digital learning among the general population [9] . For example, Ghana's national student to computer ratio stood at 42:1 in 2010 [17]. Computer literacy remains the preserve of the elite who can afford computers and opportunities to gain this education. Fortunately, there is a proliferation of smartphones in Africa. According to research firm Ovum, Africa's smartphone penetration rate will grow at 52.9% year-on-year [15]. In 2016, there were 293.8 million smartphone users and there is projected to be 929.9 million smartphones by the year 2021 [15]. Hence, smartphones provide a unique means to provide coding skills to the youth of Africa. Additionally, coding on a phone gives a huge opportunity to be very innovative with reference to teaching students how to code. For example, the coding curriculum can be designed around students building mobile-first programs such as games, which has been shown to be a more exciting and effective way to introduce programming[3, 12].

In this paper, we describe our experience running a pilot of SuaCode, an online smartphone-based coding course to teach programming fundamentals to a cohort of primarily Ghanaian students (Figure 1). In our previous work, we introduced students to coding in-person using smartphones (a first of its kind in Ghana) [5]. Building upon that work, we sought to assess the feasibility of students learning remotely using smartphones to scale the reach and impact of our course. Therefore, we gathered students' feedback in this study to assess this approach. This experience report describes the process of creating and running the course along with a discussion of lessons learned. The rest of the paper is organized as follows: we give an overview of our coding curriculum, describe the pilot experience, followed by the findings and discussion, related work and conclusion.

2 OVERVIEW OF SUACODE

The pilot of SuaCode was implemented as an 8-week online course for high school and college students in Africa by Nsesa Foundation, an education non-profit in Ghana. The course ran from from May to June, 2018. We invited high school and college students all over Ghana to apply to take part in the SuaCode pilot via advertisements on social media (Figure 2).



```
    rightPaddleY = height-paddleHeight;
  }
}

if (xBall + diameter/2 < 0)
{
  rightScore = rightScore [illegible] 1;
  xBall = width/2;
}
else if (xBall - diameter/2 > width)
```

Figure 1: Snapshot of Code on Phone

Students were asked to respond to the question "Why do you want to take part in the SuaCode pilot?" The response to this question was used to select students. Specifically, two people rated the responses on a scale of one to five and the average of the two ratings were used to select the students. A total of 117 applications were received, out of which 30 students were selected based on their motivation essays. The 30 students were all Ghanaian except one student who was from Ethiopia with (76%) being college students and 50% being female.

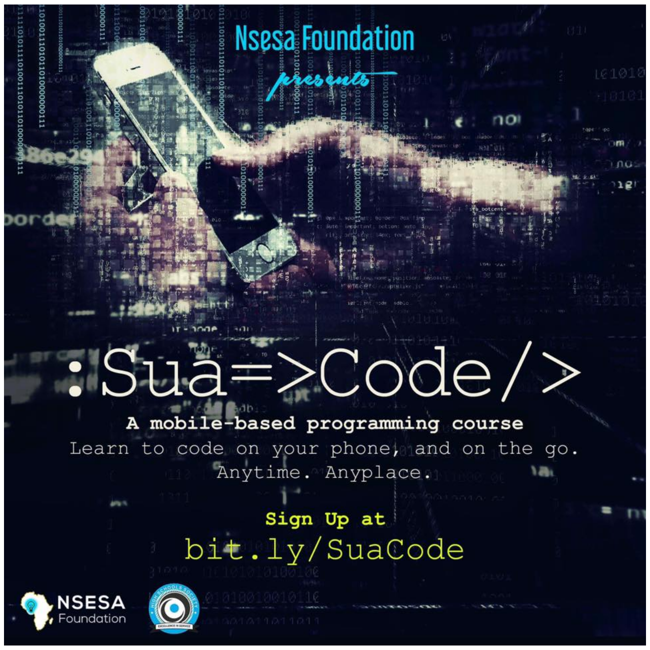


Figure 2: Poster used for advertising the SuaCode program

3 OVERVIEW OF THE CODING CURRICULUM

We designed the curriculum of our custom smartphone-based course to introduce fundamental programming concepts to students with little or no programming experience. Our curriculum was previously used in an in-person coding course in which there was a significant improvement in perceived understanding of programming concepts. The curriculum was inspired primarily by Dartmouth College’s Introduction to Programming course (CS 1) [6]. We also based our curriculum on Dartmouth’s computer science course, "Programming for Interactive Audio-Visual Arts" [7], and "the Coding Train" a set of programming tutorials, both of which use Processing for instruction [19].

The objectives of our SuaCode course were as follows:

(1) Introduce students to fundamental programming concepts
(2) Develop critical thinking and problem-solving skills.

Our course teaches programming fundamentals in a visual, interactive and fun way through game development using the Processing programming language [18] on the smartphone-based programming environments APDE for Android [2] or Processing iCompiler for iOS [8] with which to run the code on phones. Processing is an open-source, Java-based language which we chose to use because it enables learning of programming in a fun way since it can be used to easily create visual and interactive programs.

We also chose to use a game development paradigm because several studies have shown that it is an engaging, motivating and effective way to teach programming [3, 4, 11, 12]. Specifically, the course was structured such that students will build a pong game at the end of the course (Figure 3). This game has 2 paddles, one for each player and a ball. Once the ball starts moving, each player has one goal; to prevent the ball from exiting the vertical wall on his side. If the latter happens, the opponent’s score increases.

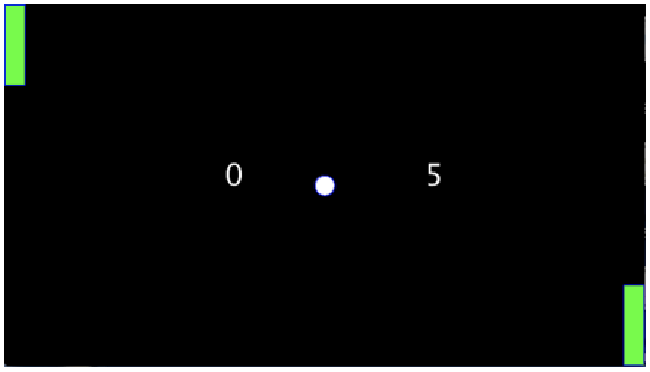


Figure 3: Pong game

Our curriculum consists of 6 lessons and it is divided into two parts. The first part introduces basic programming concepts (such as variables and functions) and the second introduces more advanced concepts (such as classes and objects). Each topic has a corresponding lesson note containing programming exercises and an end-of-lesson assignment, which incrementally built a component of the game. This arrangement was chosen to make the progression through the course more engaging. The contents of the lessons and assignments are openly available for others to use here: [1]

Completion of "Part 1" results in the building of the pong game. Part 1 consists of lessons 1, 2, 3, and 4. Lesson 1 introduces students to the Processing language and gives examples of various programs that have been created with it. The students learn how to draw and color various shapes such as line, ellipse and rectangle and fill them with colors. The assignment at the end creates an interface of the pong game consisting of different shapes and colors. Lesson 2 introduces students to variables, enabling them to move objects on the screen. At the end of the lesson, the assignment makes use of variables to represent the shapes in the pong interface from the previous assignment and then make the ball move. Lesson 3 introduces students to conditionals, enabling various actions to be performed if certain conditions are not met. At the end of the lesson, the assignment was to build upon the previous assignment and make the ball bounce if it hits the top and bottom walls of the game environment. Lesson 4 introduces students to functions, enabling them to reuse blocks of code without rewriting them. To practice, students were asked to take the previous assignment and reorganize various blocks of code with functions, and then to write functions to move the 2 paddles and also make the ball bounce off them.

Completion of "Part 2" results in adding more features to the pong game such as having several balls on the screen to make the game more interesting, or optionally adding obstacles. Lesson 5 introduces students to classes, objects and object oriented programming (OOP) enabling them to write code in a much cleaner and more intuitive manner. The lesson's assignment entailed reorganizing assignment 4 using classes and objects and including an additional ball in the game. Lesson 6 concludes the course and introduces students to loops and arrays, enabling them to easily manipulate several objects. The corresponding assignment was to build upon assignment 5 and add at least 5 balls to game. Students at this point had the options to add several features to the game to make it more interesting such as increasing the speed of the ball slightly when it bounces off the paddle, changing the color of the ball randomly as it moves or when it bounces of the paddle, varying the size of the ball from its size to zero as it moves, etc.

Table 1: Programming Curriculum

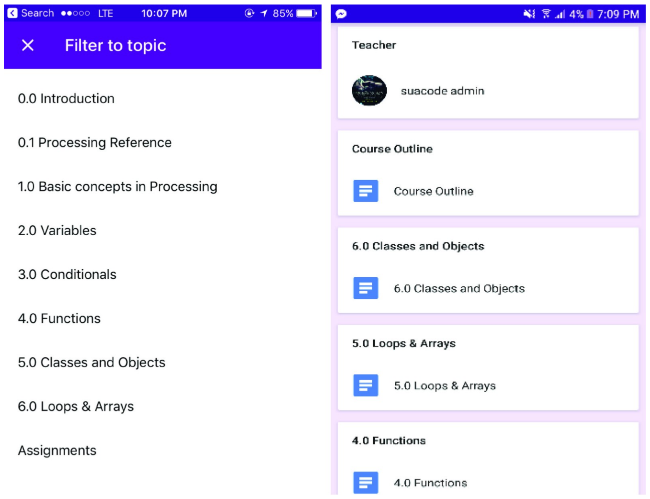
Lesson	Topic	Assignment
	PART 1	
1	0.0 Introduction 1.0 Basic Concepts in Processing	Make Interface
2	2.0 Variables	Move Ball
3	3.0 Conditionals	Bounce Ball
4	4.0 Functions	Move Paddles
	PART 2	
5	5.0 Classes and Objects	Add Extra Ball
6	6.0 Loops and Arrays	Add More Balls

[1]https://github.com/Suacode-app/Suacode/blob/master/README.md

4 COURSE LOGISTICS AND EXPERIENCE

We hosted and delivered the course online via Google classroom, a free learning management system (LMS) (Figure 3). We chose to use Google classroom among several options which we considered and tried out such as Moodle, Teachable etc. because it is the only LMS that satisfied these key requirements:

(1) Free to use both for teachers and students
(2) Course notes are accessible offline
(3) Fully functional Android and iOS apps
(4) No need to set up our server, database etc.
(5) Assignments can be assigned grades
(6) Simple to use



Search ●●○○○ LTE 10:07 PM 85%

Filter to topic

0.0 Introduction
0.1 Processing Reference
1.0 Basic concepts in Processing
2.0 Variables
3.0 Conditionals
4.0 Functions
5.0 Classes and Objects
6.0 Loops & Arrays
Assignments

4% 7:09 PM

Teacher
suacode admin

Course Outline
Course Outline

6.0 Classes and Objects
6.0 Classes and Objects

5.0 Loops & Arrays
5.0 Loops & Arrays

4.0 Functions
4.0 Functions

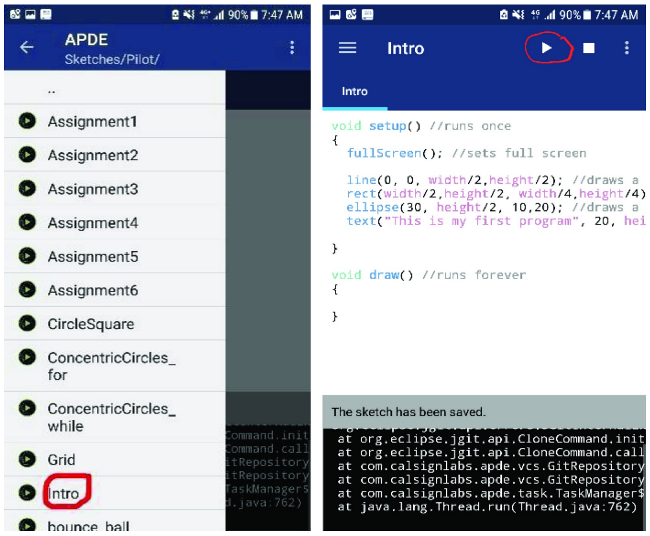
Figure 4: Snapshot of Course

For each lesson, the corresponding lesson note and assignment were made available as Google docs which were accessible in Google Classroom (Figure 4). Each week, students read the week's lesson note and also completed the corresponding assignment. Students read the lesson notes using the Google classroom iOS and Android apps. They then wrote their code for the assignments using APDE (Figure 4) or Processing iCompiler apps. We did not use videos as instruction materials but only lesson notes because Internet data is very expensive in Africa, and more so for our target population [1], students living in Africa.

Students posted questions in Google Classroom. We had two facilitators for the this pilot that answered students' questions as well as graded their assignments gave feedback for improvement. For each assignment submission, the students included a reflection essay describing among several things whether that week's lesson and it's corresponding assignment was fun, challenging, etc. They also described their experience coding the assignment with their smartphones.

5 FINDINGS AND DISCUSSION

One of the main challenges we had was that students would submit their code for the assignments without checking if it met all the specifications outlined for the assignment. Because we were



```
void setup() //runs once
{
  fullScreen(); //sets full screen

  line(0, 0, width/2,height/2); //draws a
  rect(width/2,height/2, width/4,height/4)
  ellipse(30, height/2, 10,20); //draws a
  text("This is my first program", 20, hei

}

void draw() //runs forever
{

}
```

```
 at org.eclipse.jgit.api.CloneCommand.init
 at org.eclipse.jgit.api.CloneCommand.call
 at com.calsignlabs.apde.vcs.GitRepository
 at com.calsignlabs.apde.vcs.GitRepository
 at com.calsignlabs.apde.task.TaskManager$
 at java.lang.Thread.run(Thread.java:762)
```

Figure 5: Snapshot of APDE Coding Environment

concerned with competency-based learning in which students are expected to master concepts before moving to the next module, we ended up giving feedback on their code with the opportunity to resubmit. This back and forth went on generally for about three to four times for the assignments before the students' code met all the necessary specifications. This process was very labor intensive, but we felt it was necessary to ensure they understood all the concepts in each lesson. Because of this experience, we are now exploring an automatic approach for giving feedback on assignments, and plan to develop that for subsequent iterations of the course. We intend to develop some algorithms and code that will automatically review the assignment submissions of students and give feedback about the changes needed to be made, especially if their code submissions do not meet all the requirements. This solution when implemented will aid in scaling the program to more students as we plan to run more pilots.

Additionally, out of the 30 students who were selected and invited to take the course, we had 11 students making submissions for the first assignment, and mainly 7 students consistently completing the assignments. Specifically, the 7 students completed part 1 of the course (consisting of four assignments), resulting in the building of a fully functional pong game. Unfortunately, only 1 student completed assignment 5 and no student completed the second part of the course (consisting of two assignments), which sought to teach more advanced programming concepts. For those 7 students, that completed the first part of the course, it took two months rather than the previously planned one-month period. For this pilot, we use completion of part 1 as the metric for completing the course since students built the intended pong game after completing it and also we offered part 2 as an optional next step. That puts our completion rate of 23% which is above the completion rates of online course which are mostly under 10% and on average 6.5% [10]. Nonetheless, we would like to improve on the completion rates.

We sent an email to find out the reason students were not completing assignments. It ended up being that most students were taking their examinations during the time the pilot was running, making it difficult to dedicate time to completing the assignments. This insight was useful because it showed that it is necessary to make sure the program does not run during the examination period. Additionally, in the future, we plan to have mentors assigned to students to check up on them especially when they are falling behind schedule. Also, our 1-month estimate for the completion was an underestimate of the time needed to complete the course. Hence, for future sessions, the deadlines of the assignments will be planned around an 8-week time period.

Also, there were a few issues with the APDE app. Some students reported that the app crashed sometimes and they lost their code. Debugging the issue revealed that it was happening on phones running on older Android versions. We have been in touch with the developer of the APDE app and collaborated with him to fix issues that were encountered during the program. We plan on advising our students to keep a backup of their code in a Google doc for worst case scenarios like this. Also, when code is ran, the APDE app builds and installs an Android apk which takes a long time and takes up memory for the phone since several "apps" are installed with every code run. We communicated this issue with the APDE app developer and a preview mode which avoids this process and runs the code in a much shorter time has been built into the newly updated version of the app.

Additionally, students reported that coding on a phone was a bit challenging because of the small screen size, but it was nonetheless an exciting and enjoyable experience mainly because of the convenience and accessibility it offered. Also, they said it got easier over time. This summary can be seen from the following excerpts from the reflection essays on the coding experience:

- *"Coding on my phone was very fun and convenient since I had my phone all the time and I could easily work on my code."*
- *"It is very exciting coding on phone but sometimes very challenging because of the screen size."*
- *"One problem that I have identified is not being able to have a wider screen to work on. This would have made it easier for me to see the code in its entirety. This might just be a matter of getting used to using my phone to work on my assignments."*
- *"It was really convenient, honestly. I didn't have to necessarily sit behind a desk to do it so I could do it when I was on my bed, eating, even using the bathroom. So it was fun and convenient coding on my phone."*
- *"It wasn't very different from doing it on a computer. In fact, it was more convenient as I didn't have to worry about carrying a computer everywhere."*
- *"It was really cool actually. Initially, it was not comfortable because I am used to working on my PC. However, once I started working on it, I released that it was very convenient since I can work anywhere even when I am taking a taxi, at home, work anywhere. So it was really cool. It is much better than being on social media."*
- *"I have gotten more used us my phone to code so it is no longer a difficulty."*

- *"In terms of writing the code, I found it easier than in a laptop because I could type faster. However, the small screen size doesn't allow one to view a large portion of the code at once so there's a lot of scrolling going on and that could be distracting."*
- *"This has been a very exciting experience, and I look forward to more challenging assignments."*
- *"So while reviewing the material, I found it quite interesting and easy to understand. I'm really learning a lot! And I love this course."*
- *"The material was detailed enough and easy for me to understand with a good outline. Thank you very much for this opportunity to write my first code."*

Finally, we were successful in developing students' critical thinking, problem solving, and basic coding skills as evidenced by students completing the assignments in part 1 of the course which resulted in them building a fully fledged and functioning pong game. Also, the acquiring of these skills were confirmed in the following feedback from three of the students about the course:

- *"Suacode has been a very great experience for me. I got to learn processing and actually code on my phone. I also had help from the tutors and my fellow course mates which made it easier. I learnt a lot and I'm glad I had the opportunity to be part of the first batch of suacode initiative"*
- *"I'd also like to say I'm very glad I participated in the course, it gave me a good head start in Programming which helped me a lot in my intro programming class. The course was phenomenal!"*
- *"SuaCode helped improve my algorithmic thought process. I had lots of practice with thinking in a step by step process and working through challenges."*

6 RELATED WORK

There have been a number of previous works that sought to introduce students to fundamental coding concepts via mobile programming [14, 16, 20]. Nonetheless, our approach is unique for two reasons. First, we introduce students with no or limited prior experience to programming using the smartphone as a coding environment. Secondly, we deliver our course online rather than in-person. Therefore our approach presents an interesting opportunity to scale the reach and impact of mobile programming initiatives.

In a study by Tillman et al. [20], a mobile programming environment, TouchDevelop was developed to introduce middle and high school students to programming. However, TouchDevelop was designed for Windows Phones. This is a limitation since 87.5% of smartphones worldwide operate on the Android OS [21] and also as our survey revealed that 90% of our study participants used Android phones. Our work, on the other hand, utilizes apps on iOS and Android which allow students with these smartphones to code in our course which is a useful advantage. In the same study, the authors also noted that in programming on a smartphone, actions that require fine navigation such as making structural corrections are awkward on a touchscreen where precision is limited by the size of a finger [20]. This is a challenge which students in our program also reported. Other limitations such as the smaller screen, limited battery life and relatively slower processors affect the smartphone-based programming experience [5, 20]. These challenges will need to be addressed to improve the learning experience of smartphone users.

In another study [16], the authors developed a scaffolding to support programming by university students in Kenya and South Africa who were taking an introductory course in programming using Java. In contrast to our approach, the participants in this study used desktop computers in their classroom learning and only transitioned to a mobile programming interface during the experiments. Thus it did not capture the true experience of learning programming on a phone. Besides, Mahmoud and Popowicz advocate the use of mobile phones to teach introductory programming to computer science students [14] and our approach serves this purpose.

Another innovation in our program is the delivery of the course remotely via an online system. A study by Wang [22] identified that lack of instant instructor feedback as one of the key challenges to learning programming online. We tried to mitigate this by having facilitators provide feedback to students on assignments during the course. Wang further recommends two solutions, among others, to improve the online learning experience. First, by adding multimedia content such as videos to make the instruction more engaging and understandable. Second, to create a sense of community among students to help them support each other. These suggestions can be found in our pilot in various forms. We do not include videos in our materials as stated earlier because Internet data is expensive in Africa [1]. Nonetheless, our lesson notes contain images where necessary to better explain concepts which is the next best option. Also, we encouraged students to help other students and we saw that happening but not as strongly as we hoped. We plan to encourage that more in subsequent runs of the program.

7 CONCLUSION

In this work, we described our experience delivering a smartphone-based coding course online to 30 students primarily based in Ghana. Seven of the 30 students completed the first part of the course. The reflection essays from our students showed that they enjoyed the course. Additionally, the seven students built the pong game and coding on the smartphone was not a hindrance. However, improvements such as having more mentors for the students and automated feedback will significantly improve the quality of the course and potentially improve the course completion rate. Overall, the course met the objectives of introducing students to fundamental coding concepts, critical thinking and problem solving skills. We are excited about the results of this pilot and we see the potential to scale the course to eventually bring coding skills within arm's reach of millions across Africa, literally into their palms thereby bridging Africa's digital divide.

ACKNOWLEDGMENTS

We are grateful to the Processing Foundation for supporting this work via the Processing Foundation Fellowship program through which George Boateng ran this pilot program.

REFERENCES

[1] Quartz Africa. 2019. *The cost of internet access is dropping globally but not fast enough in Africa.* Retrieved June, 2019 from https://qz.com/africa/1577429/how-much-is-1gb-of-mobile-data-in-africa/

[2] APDE [n.d.]. *APDE - Android Processing IDE.* Retrieved Jan, 2019 from https://play.google.com/store/apps/details?id=com.calsignlabs.apde

[3] Tiffany Barnes, Eve Powell, Amanda Chaffin, and Heather Lipford. 2008. Game2Learn: Improving the Motivation of CS1 Students. In *Proceedings of the 3rd International Conference on Game Development in Computer Science Education (GDCSE '08)*. ACM, New York, NY, USA, 1–5. https://doi.org/10.1145/1463673.1463674

[4] Jessica D. Bayliss and Sean Strout. 2006. Games As a "Flavor" of CS1. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '06)*. ACM, New York, NY, USA, 500–504. https://doi.org/10.1145/1121341.1121498

[5] George Boateng and Victor Kumbol. 2018. Project iSWEST: Promoting a culture of innovation in Africa through STEM. In *2018 IEEE Integrated STEM Education Conference (ISEC)*. 104–111. https://doi.org/10.1109/ISECon.2018.8340459

[6] CS1 [n.d.]. *Dartmouth CS 1.* Retrieved Jan, 2019 from http://www.cs.dartmouth.edu/~cs1

[7] CS2 [n.d.]. *Programming for Interactive Audio-Visual Arts.* Retrieved Jan, 2019 from http://aum.dartmouth.edu/~mcasey/cs2/

[8] iCompiler [n.d.]. *Processing iCompiler.* Retrieved Jan, 2019 from https://itunes.apple.com/us/app/processing-icompiler/id648955851?mt=8

[9] ITU 2018. *ITU releases 2018 global and regional ICT estimates.* Retrieved Jan, 2019 from https://www.itu.int/en/ITU-D/Statistics/Pages/stat/default.aspx

[10] Katy Jordan. 2014. Initial trends in enrolment and completion of massive open online courses. *The International Review of Research in Open and Distributed Learning* 15, 1 (Jan. 2014). https://doi.org/10.19173/irrodl.v15i1.1651

[11] Stan Kurkovsky. 2009. Engaging Students Through Mobile Game Development. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education (SIGCSE '09)*. ACM, New York, NY, USA, 44–48. https://doi.org/10.1145/1508865.1508881

[12] Scott Leutenegger and Jeffrey Edgington. 2007. A Games First Approach to Teaching Introductory Programming. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '07)*. ACM, New York, NY, USA, 115–118. https://doi.org/10.1145/1227310.1227352

[13] Digitalist Magazine. 2015. *Africa Skills Gap: Rising To Meet The Digital Challenge.* Retrieved Dec, 2017 from http://www.digitalistmag.com/improving-lives/2015/08/19/africa-skills-gap-meet-digital-challenge-03292180

[14] Qusay H. Mahmoud and Pawel Popowicz. 2010. A mobile application development approach to teaching introductory programming. In *2010 IEEE Frontiers in Education Conference (FIE)*, Vol. 00. T4F–1–T4F–6. https://doi.org/10.1109/FIE.2010.5673608

[15] Vincent Matinde. 2016. *Africa 2017: Smartphone penetration, Open Data and less online freedom,.* Retrieved Jan, 2019 from https://www.idgconnect.com/idgconnect/opinion/1022805/africa-2017-smartphone-penetration-online-freedom

[16] Chao Mbogo, Edwin Blake, and Hussein Suleman. 2016. Design and Use of Static Scaffolding Techniques to Support Java Programming on a Mobile Phone. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '16)*. ACM, New York, NY, USA, 314–319. https://doi.org/10.1145/2899415.2899456

[17] Ghana Ministry of Education. 2010. Education Sector Performance Report 2010.

[18] Processing [n.d.]. *Processing Foundation.* Retrieved Jan, 2019 from https://processing.org/

[19] The Coding Train [n.d.]. The Coding Train. Retrieved Jan, 2019 from http://thecodingtrain.com/

[20] Nikolai Tillmann, Michal Moskal, Jonathan de Halleux, Manuel Fahndrich, Judith Bishop, Arjmand Samuel, and Tao Xie. 2012. The Future of Teaching Programming is on Mobile Devices. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE '12)*. ACM, New York, NY, USA, 156–161. https://doi.org/10.1145/2325296.2325336

[21] Derek Walter. 2016. *Report: Nearly 90 percent of smartphones worldwide run Android.* Retrieved Jan, 2019 from https://www.greenbot.com/article/3138394/android/report-nearly-90-percent-of-smartphones-worldwide-run-android.html

[22] Wendy Wang. 2011. Teaching programming online. International conference on the future of education. In *Internation Conference on Future of Education.* Florence, Italy.