Motivating K-12 students learning fundamental Computer Science concepts with App Inventor

Mobile application development in secondary education

Giorgos Chatzinikolakis
Open University of Cyprus &
Greek Ministry of Education and Religious Affairs
Chios, Greece
gchatz@sch.gr

Spyros Papadakis
Open University of Cyprus &
Hellenic Open University
Patras, Greece
papadakis@eap.gr

***Abstract*—A study was conducted on the pedagogical use of App Inventor (AI) - an open-source web application that allows novices to create software applications for the Android platform- to support students in developing fundamental Computer Science (CS) concepts and critical thinking skills. A total of 35 students from four different secondary classes in Chios Island, Greece, were invited to participate in the study. In a 4-week trial teaching, students worked in pairs, sharing a mobile device to learn about topics in CS and how to program. The pre- and post-tests showed a statistically significant growth in students' domain knowledge and in their engagement. The semi-structured interviews with the teachers who participated showed that the students and the teachers perceived positively the effectiveness of the pedagogical designs on the use of programming of mobile devices to foster students' development of fundamental programming concepts through AI.**

Keywords—component; mobile learning; app inventor; computer science education; K-12

I. INTRODUCTION AND MOTIVATION

The growth of digital culture in the twenty-first century enforces the use of digital resources and new equipment in school education. Innovative educational approaches are necessary to attract high-school students to the field of computing, with such enrollment to be in decline [11, 17, 18, 19, 20].

Algorithmic and computational thinking are nowadays recognized as essential skills, necessary for all students, employees and citizens of tomorrow [2, 27, 28, 29, 30]. At the same time, both the massive penetration of mobile devices in the global market and the impact they have on new generations, can be used to facilitate teaching and learning of basic programming concepts to their users.

Smartphones and tablets may be used to motivate students to create authentic mobile applications (or simply apps) for their favorite devices, smartphones and tablets, which they use in a daily basis. These apps are easy to be showed off, as students just need to take a portable device out of their pocket and so a "viral effect" is quite possible to occur, attracting and engaging users even more, to think about, design and even develop their own fully functional apps. It is essential, for novice programmers, like K-12 students, to use an appropriate programming environment combined with attractive and meaningful educational activities.

There are many worldwide efforts to enhance learning through the use of mobile devices and Bring Your Own Device (BYOD) practice [1, 5, 10, 12], increasing the sense of commitment and productivity of students, both in and after school. However, in some countries, like Greece, the use of mobile phones is prohibited in school for both students and teachers, although everyone uses them, keeping them discreetly hidden.

To the best of our knowledge, there is no other report in Greece, until now, concerning the use of mobile devices by students about teaching and learning basic programming concepts in secondary education.

This paper presents our experience and students' feedback on teaching fundamental Computer Science (CS) concepts via App Inventor and mobile apps development, to motivate students to learn computing. The paper is organized as follows: Section II elaborates on key issues in the students' learning of CS topics and the use of mobile devices for teaching and learning programming. Section III describes and summarizes the AI environment and its potential usage in K-12 education. In Section IV, we describe the methodology of our study. Results and discussion are presented in Section V. Section VI concludes the paper and proposes future work.

II. CS TOPICS LEARNING VIA MOBILE APPS AND AI

A. CS Learning by Students

In 1961, Alan Perlis made the argument that computer science should be considered part of a liberal education, and that everyone should learn to program [9]. Programming is a fundamental component of Computer Science and a "helper" for understanding and developing computational thinking skills. According to Wing [27, 28], pRogramming should be considered as the fourth R of the basic literacy and education, together with Reading, wRiting and aRithmetic.

Students today, as digital natives [22], are capable users of technology, but their digital literacy is usually limited to just using web and office tools, so they end up being consumers of

978-1-4799-4743-0/14/$31.00 ©2014 IEEE

pre-made products, rather than understanding how all the above work is made so as to be creative themselves and, simultaneously, constructors of their own future. It is important that students learn to think, design and invent their own artifacts and express their ideas, which can be achieved by developing computational thinking skills and applying them to programming.

However, algorithmics and programming are considered by many to be a boring, technical and difficult subject, not appropriate for the average student. The extensive use of mathematical or theoretical problems, the need to write code, after learning a general purpose language and the expected syntax errors that occur, the lack of optical feedback, all seem to discourage students from choosing related lessons and CS schools [8, 23, 24]. Students' motivation to enroll in computing fields is in decline, with students finding foundational computing courses uninteresting, in contrast with the demand for professionals in the Science, Technology, Engineering and Mathematics (STEM) area [16, 19, 25].

B. Programming and Mobile Devices

According to the Constructivist learning theory by Papert [21], knowledge and learning are considered to rely on the context they take place on [3], enriched with activities dealing with students' interests and prior experience, as well as on learning-by-doing, involving the creation of meaningful artifacts, as mobile apps are. Apps' development is so ubiquitous in students' life that it becomes a powerful motivator. Via the design and development of mobile apps, students are motivated and challenged to deal with events, logical operators and expressions, variables, procedures, sequence, selection and repetition structures, as well as data structures, in order their goals to be accomplished. In this way, smart mobile devices shall stop being considered as black boxes by students and young people shall be given a chance to evolve into creators and shapers of their own future, instead of plain consumers of technology.

The usage of the latest media and technology is more motivating than the old and traditional ones [8]. There is a need for exposing students to computing in a more engaging way. A common way to address this challenge is to lower the barrier of initial entry to programming through the use of visual programming tools like Scratch and Alice [23, 24].

In Greece, curricula for introductory programming courses in primary and secondary education, recommend the use of LOGO-like visual programming environments, especially for the youngest students, through a spiral approach on CS subjects and the use of multiple software tools. On the field of programming, Scratch and Logo-Like environments like EasyLogo and MicroWorlds Pro are recommended for use in primary education, gradually moving to more complex environments like Kodu, Alice, Snap! (formerly BYOB) and Gamemaker in secondary education. Educational block-based graphical programming languages enable students to create interactive stories, animations, games, and more, while learning about mathematical and computational ideas. Most of these environments may be used as multimedia authoring tools by students, scholars, teachers, and parents for a range of educational and entertainment constructivist purposes, from simulations and visualizations of experiments, recording lectures with animated presentations, to social sciences animated stories, and interactive art and music.

The latest New Media Consortium annual reports (NMC Horizon Reports K-12 Editions) describe the great potential and the impact mobile devices may have on education [13, 14, 15]. Mobility, computational power, a great variety of sensors and connectivity options are some of the affordances of popular smart mobile devices making them a great complementary of the desktop computers traditionally used in Greek school Computer labs.

According to Holz [11], mobile phones are the only device equally used by female and male students, with female students to be slightly ahead. The results of 2009 "EU Kids Online II" research program for Greece [32] showed that among 10-14 years old kids, 43% of boys and 51% of girls own a mobile phone, while in the age group of 15-18, the percentages rise up to 57% for boys and 64% for girls.

III. APP INVENTOR FOR ANDROID IN K-12 EDUCATION

World-wide data from smartphone sales show that the Android platform is dominant in the world market [33, 34].

Required time for someone to learn native Android programming, even in CS0 level, is usually two semesters, one for learning Java and one for programming apps in Android Software Development Kit (SDK) [26]. This is unobtainable in secondary education, where students have little or no programming experience and only one or two hours per week in their class' schedule to be involved in CS related lessons.

App Inventor as an educational programming tool, on the other hand, maintains the advantages of Android platform, while reducing the negatives of steep learning curve and easing handling of complex interactions with the Android operating system, incorporating the benefits of optical programming [6]. Still, app development demands a set of necessary skills and sets a challenge for novice programmers, as it requires understanding of event-driven and object-oriented programming.

A. MIT App Inventor

App Inventor for Android is an open-source web application originally provided by Google, and now maintained by the Massachusetts Institute of Technology (MIT).

The educational perspective that motivates AI holds that programming can be a vehicle for engaging powerful ideas through active learning. With AI, students can create a fully-functional Android application that can do almost anything a traditional Java app can do, and afterwards the app can be installed to phones or even published in the Android market [29].

AI allows newcomers to computer programming to create software applications for the Android operating system (OS) and mobile devices. It uses a graphical interface, very similar to Scratch and the StarLogo TNG user interface, which allows users to drag-and-drop visual objects to create an application that can run on Android devices. This block based

programming tool provides built-in components that handle complex interactions with the Android operating system, featuring a graphical interface. Various objects are linked together to create an app, mainly by drag and drop operations. It reduces the cognitive load required for code to be written in a strict programming language and may act as a bridge (stepping stone) to Java programming and Android SDK [25]. It is easy and fun to use, helps focusing on design and implementation of the solution, without the fear of syntax errors and language specific notation, in a way similar to Scratch [3, 23, 24]. It allows students to focus on the most interesting and meaningful aspects of programming, that is, designing and developing solutions for their own ideas, building mobile apps that are useful and fun to share, and feeling creative and proud of their creations/artifacts.

The App Inventor web-based application was firstly made available to the public in 2010, as a beta project of Google Labs. In 2011, Google terminated the project, which was acquired by Massachusetts Institute of Technology (MIT) Center for Mobile Learning (http://mitmobilelearning.org/), which released it in 2012, as an open-source software and has been maintaining and developing it since then. In December 2013, together with the beginning of the Hour of Code (a nationwide initiative by Computer Science Education Week and Code.org to introduce millions of students to one hour of computer science and computer programming) the second version of AI (AI2) was launched (http://appinventor.mit.edu).

The first version of App Inventor required a program to be installed on computers for the block editor and android emulator, and had a mostly web-based interface, integrated with the client application, so, for some users, administrative privileges might have been required for its installation. However, AI2 features a full web interface that includes block editing, which eliminates the need for the client program, as AI2 replaced Java Open Blocks library with Blockly, a blocks editor that runs within the browser.

App design and development takes place entirely on the user's web browser. AI2 environment is divided into two tabs: the Design Editor and the Blocks Editor. In the Design Editor tab, users design the app interface by dragging components in the Viewer window and changing their properties. In the Blocks Editor tab, users program the object's behavior on various events by putting together puzzle-piece blocks. Code blocks can only be snapped together in ways that make sense, preventing students from using them in invalid combinations, enforcing proper programming syntax and ensuring that novice programmers learn the proper way to assemble and formulate programming logic in their scripts.

Moreover, the AI2 Companion app, available in Google Play, enables real-time debugging on connected devices via Wi-Fi, not just USB (Fig. 1). There is also an Android emulator available that can be installed on users' computers, but a virtual machine cannot emulate all functions of a real device, so the user experience is inevitably a less accurate depiction of the mobile user experience. The Android emulator option eliminates - or at least reduces - the need for mobile devices in class, although such devices are continuously becoming more affordable in price and they provide the benefit of a full hands-on experience.

Fig. 1. A pair of students from the workshop, preparing to test their app.

AI support for "live" testing on all stages of building an app, promotes students' experiments in the Design and Blocks editors and allows the direct view of the results of their actions on their mobile device's screen. This immediate feedback helps in debugging, experimenting and learning by doing, without the fear of breaking things out. It is also time-saving and helps students evaluate their work in progress and reflect on their goal and how to achieve it.

B. Conclusions on the Use of AI in Other Countries

In our knowledge, there are no other reports on AI use in secondary education in Greece. However, in other countries, the most recent research [16, 17, 18, 25] found that AI promotes problem solving based learning, providing a low floor-high ceiling environment with wide walls, it combines event driven and object oriented programming, along with classic programming structures and it gives extra motives to students because of the mobility and the practical use of the created apps [5, 19, 20]. Additionally, support from MIT and Google is praised as it offers great advantages and a sense of safety to teachers and students.

AI as a teaching platform, encourages students to focus on problem solving rather than coding syntax, while providing a more "hands-on" experience, compared to the fantasy worlds of other virtual environments. Morelli [20] specifically mentions the "viral effect" AI may have on students, easily showing off their fun-to-share apps to friends and others, spreading the word that dealing with programming can be quite interesting.

Robotics and video game creation have been cited as suitable settings for children's introduction to essential CS concepts [23]. However, Grover [7] found AI to be the most gender neutral and truly "democratic" tool among others, as it doesn't appeal more to male interests.

Dabney [4] and Roy [25] both conclude that AI presents a useful and smooth transition to developing mobile apps in Java using the Android SDK, once students gain sufficient experience in programming logic.

C. AI and CS Curricula in Greece

According to the latest primary and secondary education CS curricula in Greece, students graduating from the Primary school and Gymnasium are expected to have already used Scratch, so they will easily get familiar with AI in Lyceum. AI is considered to be similar to Scratch, but for smart mobile devices, which gives the bonus of dealing with real world problems and "portable" creations. The learning activities and examples are not limited to virtual worlds but may have actual meaning in everyday life, so students' inspiration can easily be shaped into authentic apps. So, AI provides an extended hands-on experience, not limited to the imaginary worlds of Alice or Scratch.

AI supports all programming fundamental concepts, from variables to data structures like lists, and even concurrency, through the use of multiple clock/timer components.

A recent report in Greece [31] on the conditions which a programming environment should meet in order to be chosen by an educator in school are: a) meets the objectives of curriculum, b) belongs to a group of software with common philosophy, c) has an optical programming interface, d) is free software, e) interface is available in Greek language, and f) there is a learning community focused on the lesson's goals. AI already satisfies the first four criteria. Due to its free and open source character and its growing rate of development, we expect that even the last two criteria will be satisfied soon.

Furthermore, in 2014, a new curriculum was announced by the Greek Ministry of Education and Religious Affairs, to be applied in the coming year for the first time, which has added App Inventor in the recommended programming tools on the first class of Lyceum targeting the specific educational goal of "*students analyzing problems, designing and developing apps for smart mobile devices*".

IV. Our Study

Our project explored the possibility and suitability of using mobile devices and AI to motivate students to learn about CS and programming fundamental concepts, as they are described in Greek CS curriculum for secondary education. Due to some practical limitations and constrains, it was impossible for the researchers to conduct the experiment in a real school class, as a part of the daily schedule. Thus, we invited students of varying ages (13-16 years old) and schools, coming from uneven programming backgrounds, to participate in two after-school workshops, arranged in collaboration with five CS teachers from Chios' Association of CS Teachers (http://www.sepchiou.gr).

A. Methodology

The research procedure combined qualitative and quantitative methods for data collection, involving researchers' direct participation in all workshop sessions, pre- and post- electronic questionnaires completed by participating students, as well as semi-structured interviews with participating teachers (in the parallel role of educator and observer). This triangulation technique improved the reliability, validity, consistence and completeness of the collected data.

B. The Case Study

The case study of experimenting with two similar groups of students, consisting a total of 35 students from Gymnasiums and General and Vocational Lyceums, was conducted in Feb - Apr 2014, in two different school computer labs in the Chios Island. Each workshop consisted of 4 two-hour sessions following a project-based learning model to introduce computational concepts to participants via AI.

Students worked in pairs and used their personal mobile devices (Android smartphones or tablets). In all cases, there was at least one compatible device per team, so the Android Emulator software was not used. Besides, both school labs offered Wi-Fi, so live testing on students' devices was very easy and only required the installation of MIT AI2 Companion and a QR code scanner app (Fig. 1).

We decided to minimize lecturing on behalf of the instructors and let students learn on their own. Students were asked to follow the given activities instructions and to form questions to the educators only after having discussed their issues with each other. Using an instructional scaffolding technique, we tried to bridge the gap between what students could do on their own and what required extra support, helping them to achieve their learning goals and develop autonomous learning strategies. Errors were welcomed and students were encouraged to freely experiment and work at their own pace, with the option of choosing which exercises/activities to complete in every book chapter.

C. Instructional Material and Digital Learning Resources

The AI's official website offered various tutorials, but none of them was available in the Greek language. Furthermore, the subjects of those tutorials did not match the exact needs of the specific topics from the curricula that were examined. At the same time, CS teachers in Greece are encouraged to produce their own learning material, based on their students' specific needs and interests. These facts led us to develop our own original activities and educational material in Greek, in the form of a book that is freely available (CC-BY-SA license) in the Chios CS Teachers Association website (http://www.sepchiou.gr/ai), along with sample multimedia files and AI source projects. All the activities we developed, were used as tools for supporting the learning needs of the workshops, focusing on specific programming topics. We tried to utilize all mobile devices' affordances, like sensors, camera, short text messages (SMS), Text-To-Speech (TTS) and to cover fundamental concepts, like events, variables, procedures, logical expressions, selection structures, loops and lists, in a total of 4 lessons within 8 hours (Table I). All the above topics are included in the Greek K-12 CS curriculum and are present in almost every class according to a spiral learning approach. Introductory activities were quite analytical, gradually becoming less detailed, whenever a topic/concept was repeated. In addition, soon after students had got familiar with the Design Editor, they were often provided with AI source files, already including necessary components and multimedia files, so as to allow them to focus on programming logic, developing, extending or making corrections to existing code blocks.

TABLE I. STRUCTURE OF WORKSHOPS LESSONS

Concepts	Device Features	Sample Activities[a]
Events, sequence of actions, randomness	Touch screen, motion sensors (shaking detection), SMS	RollTheDice, GuardDog, PanicButton
Variables, counters, logical expressions, if-then-else	Touch screen, SMS, accelerometer	HeadsOrTails, RollTheDice2, GuardDog2, TimerApp
Procedures and Applying them in various problems	Screen drag, drawing on screen, camera, orientation sensor, file saving, TTS, GPS, contacts	PhotoGallery, Calculator, FingerPainting, Compass, CatchTheFlag
For / While, Lists, TinyDB	Drawing on screen, GPS, contacts, DB storage, TTS	RandomDrops, WhereIsMyBike, PanicButton2, KidsMath, SpeakingDictionary

[a.] Some sample apps are available in Google play: http://goo.gl/eIZl82

V. RESULTS AND DISCUSSION

Our sample included 32 boys and 3 girls, 25.7% being Gymnasium students and 74.4% being Lyceum students with 80.0% of the sample owning a compatible Android device (smartphone or tablet). 50.0% of them were unsure of which programming concepts they had used in the past, while 65.0% answered that they had used a Logo-like programming tool in school and 91.0% that had never used AI before.

Pre- and post- workshop surveys' data were collected using unique codes for each participant. Apart from questions focusing on students' experience, there were also some common questions in both questionnaires. SPSS 17 was used and several statistical tests were conducted to evaluate students' answers on a Likert-based scale. Paired samples tests (a=0.05) for common questions in the two surveys indicated some statistically significant change in students' attitude to computing and programming, as in the question "*To what extent do you think programming concepts (...) relate to your everyday life?*" (p=0.017).

Students agree that AI was fun to use and helped them experiment and better understand what programming is about (Table II). When being questioned about the opposite, that is, what they didn't like in AI, they mostly agree (29.0%) that it was relatively difficult for them to find the command blocks needed, as well as they noted the lack of an Undo operation (26.0%). Regarding specific programming topics, the main difficulty was observed in students' capability to define their own variables when necessary, while they didn't face any significant difficulties in manipulating variables' values, except for the fact that they sometimes forgot to update the relevant text labels on screen with the appropriate commands.

TABLE II. STUDENTS' OPINION ON AI POSITIVE POINTS

AI helped you to …	Mean[b]	Std. Deviation
Experiment	1,34	0,684
Understand what programming is about	1,31	0,583
Build good looking apps	1,91	0,853
Avoid errors in the development phase	1,89	0,900
Have fun	1,46	0,701

[b.] Likert Scale Values: 1=Totally agree, 5= Totally disagree

Students found the mobile application programming to be more interesting than the computer programming they were used to, as there was a change in their attitude before and after the experiment (Fig. 2), from 17.0% to 37.0% for mobiles.

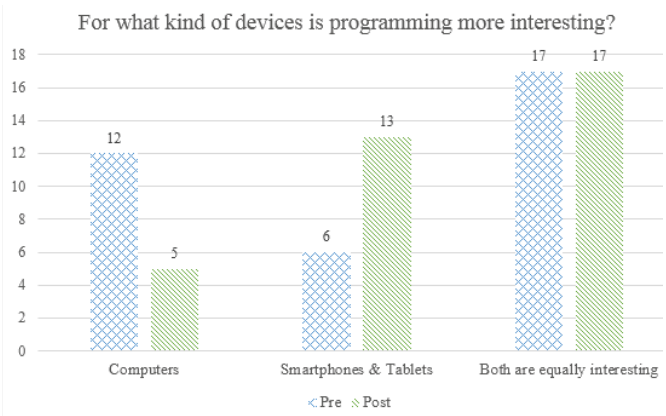

Fig. 2. Students' interest change per type of device programming.

However, students, especially the older ones, felt that AI may pose restrictions on what they can develop after a while, wondering if it was possible to transfer their apps in a so-called "real" programming environment. Students' perception of such an environment, was that of a traditional text-based one. While, there is a recent report [25] that AI projects can be successfully exported and imported to Java programs in Android SDK, this feature was absent in the latest AI2 version, nor does such an unofficial tool exist, in our knowledge, so we explained to our students that such a transition was impossible. That is the main reason why we observed a slight decrease (mean difference=-0.343) in their answers regarding whether programming in AI (post- survey question) is as useful as computer programming in general (pre- survey question).

The use of their personal mobile devices motivated and engaged students in app development, making them sufficiently use all the introduced programming concepts. Live testing capability with its direct feedback kept them excited until the end of the sessions. All students wanted to permanently install their apps on their devices and from the very first session they were asking how to share them with each other.

By the end of the 8-hour workshops, 91.0% of the participants considered themselves capable of building their own apps (Fig. 3), while 83.0% claimed that they intended to keep using AI, even after our sessions (Fig. 4).

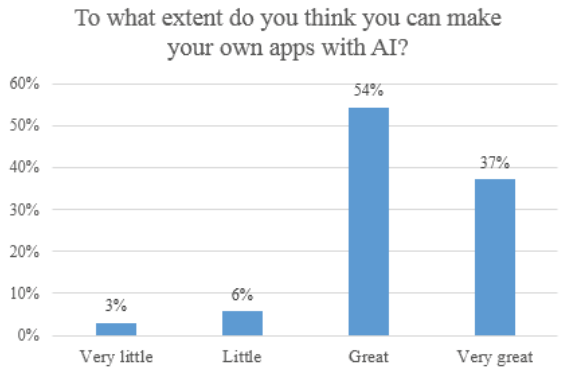

Fig. 3. Students' claimed capability of making their own apps.

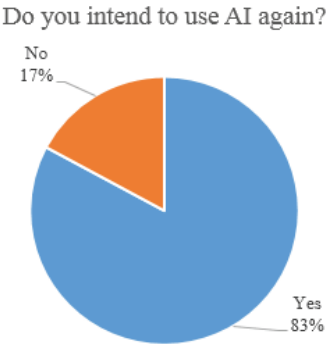

Fig. 4. Students' intention on reusing AI after the end of the workshops.

Students' feelings and their attitude towards mobile apps' development and AI may be summarized in the next two figures which speak for themselves. Fig. 5 shows that 82.0% of them consider that AI would make CS lessons in school more interesting. In the last session, we asked students to write a free thought on a Post-It note regarding their participation in the workshop. Fig. 7 shows the thought of two students – one of them was a girl – that preferred expressing themselves in AI code by combining some programming concepts they had learned.

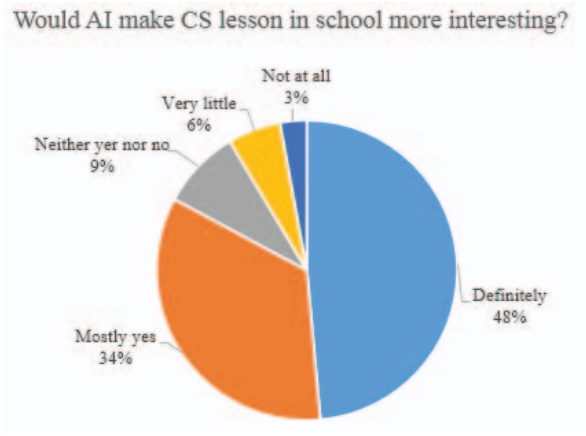

Fig. 5. Students' opinion on AI use in school.

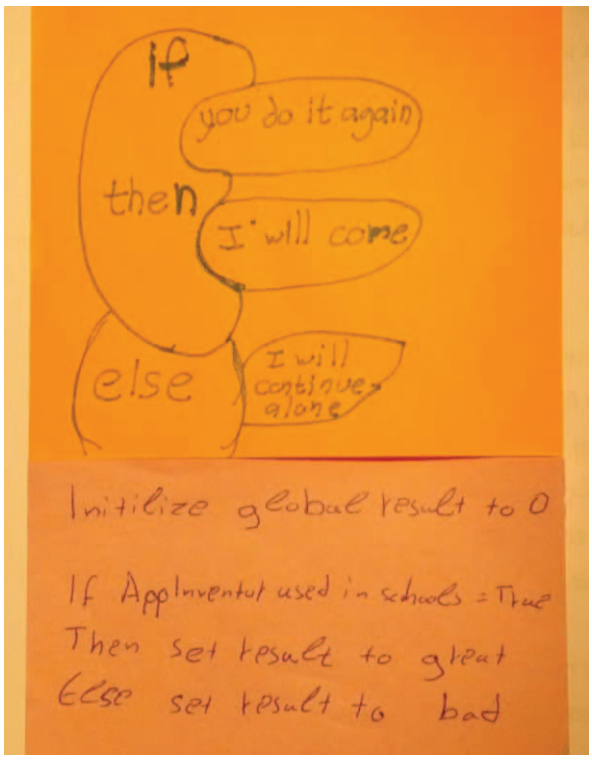

Fig. 6. Two interesting Post-It notes with students' feelings on AI .

VI. Conclusions and Future Work

Programming, despite its importance, is a difficult subject for students. Our main goal was to introduce students to fundamental computing and programming concepts and stimulate their interest in the field of computing. Novices found AI and mobile devices really attractive, forming a fun-to-work-with educational environment. Smartphones and tablets seemed to be a more interesting target device for app development compared to plain computers.

Students successfully used all examined programming concepts and they felt free to experiment and apply most of them in their own ways. What has to be mentioned was the strong tendency of participants to experiment, modify and extend their apps since the very first session. A typical example of this was the crafty modification a student made to the RollTheDice app, adding a secret button, so as to get sixs on demand, instead of totally random rolls (Fig. 7). This is a great example of students' creativity.

Compared to Scratch, AI lacks community matters, like easy project sharing, and it is not yet available in the Greek language (although English did not seem to pose a problem for our sample). Some technical issues and deficiencies were noticed too, like the absence of undo operation, object cloning or dynamic creation of them on execution, offered by Scratch. Nevertheless, MIT and users' activity indicate a great potential of AI growing and maturing in all areas.

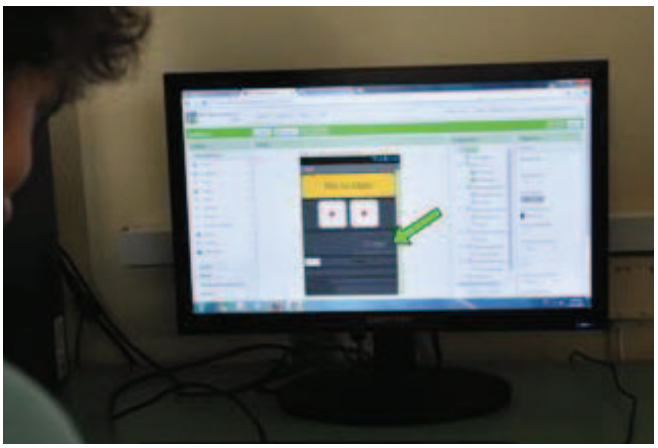

Fig. 7. A student's modification to a simple RollTheDice app.

Our conclusion is that AI and mobile devices, combined with activities focusing on real world problems and students' interests, provide an authentic platform with great potential for learning and should be seriously considered as a means to attract students to the computing field.

Especially, in Greek education, students' prior use of similar tools, like Scratch, prepares them for an easier transition to AI, so AI is recommended to be used in Lyceum classes as it meets all educational goals of the curriculum on the subject of programming.

The problem of the forbidden use of mobile devices in Greek schools and teachers' reservations may be overcome by the use of SIM-free smartphones or tablets. Nevertheless, in our sample, the use of mobile devices posed no problem in students' concentration, engaging them to work, as BYOD practice indicates.

The use of mobile devices in Greek schools for learning is an area which has not yet been thoroughly examined. Our research showed that mobile devices and AI enhanced students' attitude towards programming, but we did not examine changes in their problem-solving skills. We also did not have the time to assign student projects totally based on their interests and evaluate them. Additionally, we conducted our case study after school and not during the typical daily school schedule. Finally, our random sample was representative of ages and included students of four different classes, but due to voluntary participation and the fact that one of the workshops was conducted in a male-only boarding school, there were only 3 girls in our sessions, so we could not study the effect of mobile devices and AI in relation to students' gender.

ACKNOWLEDGMENT

We would like to thank our colleague in Chios, Mr. V. Vassilakis for our pleasant and productive collaboration in developing the educational material and conducting the workshops, as well as all members of Chios CS Teachers Association for their help and support. Finally, we thank Mrs. N. Karagianni and Mr. Y. Prodromou about their help with proofreading.

REFERENCES

[1] R. Ballagas, M. Rohs, J. G. Sheridan and J. Borchers, "Byod: Bring your own device," in Proceedings of the Workshop on Ubiquitous Display Environments, Ubicomp, 2004, vol. 2004.

[2] V. Barr and C. Stephenson, "Bringing computational thinking to K-12: what is involved and what is the role of the computer science education community?," ACM Inroads, vol. 2, no. 1, pp. 48–54, 2011.

[3] S. Cooper and S. Cunningham, "Teaching computer science in context," ACM Inroads, vol. 1, no. 1, pp. 5–8, 2010.

[4] M. Dabney, B. Dean and T. Rogers, "No sensor left behind: enriching computing education with mobile devices," in Proceeding of the 44th ACM technical symposium on Computer science education, 2013, pp. 627-632.

[5] K. Gittman, "Bringing Your Own Mobile Device to the Classroom Kimberley L. Gittman Virginia Tech," 2014.

[6] J. Gray, H. Abelson, D. Wolber, and M. Friend, "Teaching CS principles with app inventor," in Proceedings of the 50th Annual Southeast Regional Conference, 2012, pp. 405–406.

[7] S. Grover and R. Pea, "Using a discourse-intensive pedagogy and android's app inventor for introducing computational concepts to middle school students," in Proceeding of the 44th ACM technical symposium on Computer science education, 2013, pp. 723-728.

[8] M. Guzdial and E. Soloway, "Teaching the Nintendo generation to program," Communications of the ACM, vol. 45, no. 4, pp. 17–21, 2002.

[9] M. Guzdial and E. Soloway, "Computer science is more important than calculus: the challenge of living up to our potential," SIGCSE Bulletin, vol. 35, no. 2, pp. 5–8, 2003.

[10] E. Hartnell-Young and N. Heym, "How mobile phones help learning in secondary schools," A report to Becta. Copyright University of Nottingham. Online: http://schools.becta.org.uk/upload-dir/downloads/page_documents/research/lsri_report.pdf, (Accessed: 31 August 2008), 2008.

[11] J. Holz, T. Leonhardt, and U. Schroeder, "Using smartphones to motivate secondary school students for informatics," in Proceedings of the 11th Koli Calling International Conference on Computing Education Research, 2011, pp. 89–94.

[12] J. Huizenga, W. Admiraal, S. Akkerman, and G. ten Dam, "Mobile game-based learning in secondary education: engagement, motivation and learning in a mobile city game," Journal of Computer Assisted Learning, vol. 25, no. 4, pp. 332–344, 2009.

[13] L. Johnson, S. Adams, and M. Cummins, NMC horizon report: 2012 K–12 edition. The New Media Consortium Austin, TX, 2012.

[14] L. Johnson, S. Adams Becker, M. Cummins, V. Estrada, A. Freeman, and H. Ludgate, NMC Horizon Report 2013 K-12 Edition. Austin, Texas: The New Media Consortium. The New Media Consortium Austin, TX, 2013.

[15] L. Johnson, R. Smith, H. Willis, A. Levine, and K. Haywood, The 2011 Horizon Report. Austin, Texas: The New Media Consortium, 2011. The New Media Consortium Austin, TX, 2006.

[16] M. Karakus, S. Uludag, E. Guler, S. W. Turner, and A. Ugur, "Teaching computing and programming fundamentals via App Inventor for Android," in Information Technology Based Higher Education and Training (ITHET), 2012 International Conference on, 2012, pp. 1–8.

[17] S. Kurkovsky, "Engaging students through mobile game development," in ACM SIGCSE Bulletin, 2009, vol. 41, pp. 44–48.

[18] X. M. Liu and D. Murphy, "Using Mobile Apps to Entice General Education Students into Technology Fields," Information Systems Education Journal, vol. 11, no. 1, p. 25, 2013.

[19] J. Margolis and A. Fisher, Unlocking the clubhouse: Women in computing. MIT press, 2003.

[20] R. Morelli, T. de Lanerolle, P. Lake, N. Limardo, E. Tamotsu, and C. Uche, "Can android app inventor bring computational thinking to k-12," in Proc. 42nd ACM technical symposium on Computer science education (SIGCSE'11), 2011.

[21] S. Papert and I. Harel, "Situating constructionism," Constructionism, vol. 36, pp. 1–11, 1991.

[22] M. Prensky, “Digital natives, digital immigrants part 1,” On the horizon, vol. 9, no. 5, pp. 1–6, 2001.

[23] A. Repenning, D. Webb, and A. Ioannidou, “Scalable game design and the development of a checklist for getting computational thinking into public schools,” in Proceedings of the 41st ACM technical symposium on Computer science education, 2010, pp. 265–269.

[24] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and others, “Scratch: programming for all,” Communications of the ACM, vol. 52, no. 11, pp. 60–67, 2009.

[25] K. Roy, “App inventor for android: report from a summer camp,” in Proceedings of the 43rd ACM technical symposium on Computer Science Education, 2012, pp. 283–288.

[26] S. Tigrek and M. Obadat, “Teaching smartphones programming using (Android Java): Pedagogy and innovation,” in Information Technology Based Higher Education and Training (ITHET), 2012 International Conference on, 2012, pp. 1–7.

[27] J. M. Wing, “Computational thinking,” Communications of the ACM, vol. 49, no. 3, pp. 33–35, 2006.

[28] J. M. Wing, “Computational thinking.,” in VL/HCC, 2011, p. 3.

[29] D. Wolber, H. Abelson, E. Spertus, and L. Looney, App Inventor. O’Reilly Media, Inc., 2011.

[30] A. Yadav, N. Zhou, C. Mayfield, S. Hambrusch, and J. T. Korb, “Introducing computational thinking in education courses,” in Proceedings of the 42nd ACM technical symposium on Computer science education, 2011, pp. 465–470.

[31] Τ. Λαδιάς. «Ο προγραμματισμός Η/Υ στο νέο Πρόγραμμα Σπουδών της υποχρεωτικής εκπαίδευσης στο πλαίσιο του μαθήματος για τον Πληροφορικό Γραμματισμό», 2012.

[32] Λ. Τσαλίκη, Δ. Χρονάκη «Παιδιά και Διαδίκτυο στην Ελλάδα», GRKGO, 2010. http://www2.media.uoa.gr/people/tsaliki/wp-content/uploads/2010/07/Tsaliki_Kids_and_Internet_in_Greece.pdf

[33] Ofcom “Children and Parents: Use and Attitudes Report” Research Document, 3 October 2013. http://stakeholders.ofcom.org.uk/market-data-research/media-literacy-pubs/

[34] Top Smartphone Operating Systems, Shipments and Market Share, 2013 Q3. IDC Worldwide Mobile Phone Tracker, Aug. 2013. http://goo.gl/1rMD4r