DYNAMIC PROGRAM VISUALIZATION ON ANDROID SMARTPHONES FOR NOVICE JAVA PROGRAMMERS

Elhard Kumalija, Sun Yi and Ymran Fatih
Kobe Institute of Computing
〒650-0001, Chuo Ward, Kanocho, 2−2−7, Kobe, Japan

ABSTRACT

Dynamic program visualization tools helps to reduce the cognitive load of students in learning programming. However, to authors' understanding there is no dynamic program visualization tool that can be used in a widely available smartphones. In this study, we design a Dynamic visualization engine for java programs that is integrated to java code interpreter that runs on android smartphones. This tool will be valuable to students who don't have access to computers. It can also increase productivity in smartphone usage among lower secondary schools students.

KEYWORDS

M-learning, E-learning, Program Visualization, Java Programming

1. INTRODUCTION

We are surrounded by digital technology everywhere we are than ever before. Computer science literacy is now the needed basic skill as mathematics. Computer science is now being integrated in primary and secondary schools curriculum to equip students with the basic skills needed in this digital society. The computer science curriculum now puts emphasis in computation skills than previous. Visual approach to teaching computational skills through program visualization has seen as an effective way to motivate students to learn programming and reduce the cognitive load of understanding programming. Despite higher ownership of smartphones compared to personal computers there is no program visualization tool for learning programming using smartphones known to authors. Time spend by individual in using smartphones is higher than time spent in using PCs. Students are always accompanied with smartphones when commuting to school and on the way back home. More than 70% of their time is spend in social, entertainment and gaming activities while only 4% is spent on productivity (Jesse G.R., 2015). This paper presents a design of Dynamic program visualization on android smartphones for novice java programmers. The tool is expected to be a great aid to lower secondary schools students in schools where there is no computer laboratories but also to help students in schools with computer laboratories to use their smartphones in a productive way.

2. BACKGROUND

Nowadays computer science literacy is important skill for every career. Many countries are introducing compulsory computer science subjects in all levels of education. In 1997, the Ministry of Education of China emphasized importance of computing in college education regardless of discipline (Pan T.Y., 2017). In 1996 the Government of Tanzania launched computer science curriculum for secondary schools. These early initiatives were focused on skills to use computer tools. However needed computer literacy skill is shifting from learning computer tools towards equipping students with computation skills.

Every country is thriving to empower its youth in computer science and computational thinking skills. In 2016, the then President of United States America started a new initiative to empower a generation of American students with the computer science skills they need to thrive in a digital economy. The focus of computer science for all is students from kindergarten through high school to learn computer science and be

equipped with the computational thinking skills they need to be creators in the digital economy, not just consumers, and to be active citizens in our technology-driven world (Megan 2016). Basic computer skills for college students (CS0) reform happening in China that shifts the focus of the course from computer tools and skills to computational thinking (Pan T.Y., 2017). There are also different campaigns like coder dojo and hour of code that are motivating students all over the world to learn computer science skills.

2.1 Computational Literacy and Digital Divide

Despite all efforts to teach computational skills to students in primary and secondary schools the goals are still far from being attained. Moreover there is a big difference between developed and developing countries. According to United Nations statistics division, data shows that the proportion of youth and adults with information and communications technology (ICT) skills to write a computer program using any language in all countries is less than 10% but in developing countries the results are worse, with less than 1% in some countries. Moreover, the proportional of primary and secondary schools with access to computers for pedagogical purposes varies from 90% to less than 7% in other countries (Sustainable Development Goals (SDG) Indicators | United Nations Statistics Division, 2017). The low computer literacy can be attributed not only to lack of computer laboratories in primary and secondary schools in developing countries but also the low usage of computers among youth.

The low computation literacy is due to difficult in comprehension of computation and digital gap. Programming is a difficult cognitive skill to learn. Mastering the basis of a programming language is a huge problem for many students. In order to write a simple program they need to have a basic knowledge of variables, input/output of data, control structures and other areas. There are much more complex concepts such as pointers, abstraction or exception handling. Moreover, Students in resource challenged environments lack opportunity to learn computation skills due to lack of computer labs.

In SDGs goal 4 is to ensure inclusive and quality education for all and promote lifelong learning. The one of the aim is by 2030 to ensure that all girls and boys complete free, equitable and quality primary and secondary education. Recognizing that computer science is a new basic skill necessary for economic opportunity and social mobility to ensure that students get equitable and quality education the digital divide needs to be addressed.

2.2 Program Visualization on Android Smartphone

Different tools have been developed to help, motivate and make learning computation skills and programming an interesting endeavor. These tools employs visual approach to programming and can be categorized into visual programming environments like Scratch and AppInvetor and Program visualization tools for example Jeliot and Python tutor.

Program Visualization tools are promising programming teaching tools in early stages of the learning path of a programmer, teaching the students the basics of programming and algorithms. Software Visualization tries to represent software systems in ways that help the user, developer, or student to understand them (Bhattacharya, P. Et al, 2011). A study of 600 students in a programming course with focus on visual approach showed increase in pass rate from 12% to 23% (Cisar S.M. Et al. 2011). According to Bhattacharya et al (2011) visualization would help novice programmer understand many abstract concepts and how to implement the concepts easier and better.

Furthermore, there is a widespread use of smartphones and nowadays smartphones are so powerful that there is no different between a laptop and smartphone apart from screen size. Youth are accompanied with their smartphones when commuting from home to school and back, the mobility of smartphones make their usage simple and everywhere.

To address the digital divide gap, **dynamic program visualization on android smartphones for novice java programmers** is proposed. The proposed solution capitalizes on youth preference to use smartphones and advantages of visual approach to learning programming. Students in resource challenged environments will be able to write java code, compile and visualize code execution on android smartphones. The target is primary and lower secondary school students.

3. RELEVANT WORK

This work involves two aspects the first is programming using smartphones and the second is providing dynamic program visualization.

3.1 Program Visualization Tools

Program visualization is graphical presentation of code execution. Jeliot3 is a program visualization application. It visualizes how a Java program is interpreted. Method calls, variables, operation are displayed on a screen as the animation goes on, allowing the student to follow the execution of a program step by step. Programs can be created from scratch or they can be modified from previously stored code examples; all the visualization is automatically generated (Moreno, A. Et al. 2005).

mJeliot supports interactive visualization of program behavior where learners become actively involved in testing their knowledge in an environment where they receive direct feedback about their own hypotheses (Pears, et al. 2011). What is missing is program visualization tool on smartphones as all tools (known to authors) can be used with personal computers.

3.2 Smartphones in Learning Programming

The growth of mobile technologies was evolutionary in the progress of technology; it opened a revolution in computing in a quicker time frame. The easy availability and extreme mobility with rich set of applications made smartphones an inevitable tool for students.

Different approaches has been adopted to utilize the potential of mobile phone to students studying programming, example delivering education content, introduction to programming where students learn directly by developing smartphone applications and using smartphone to write programs that will be run on smartphones (John M.S. & Ran, M. 2015). There are mobile platforms for learning programming on smartphones like mobProg a mobile-based application that provides students with a smart phone based platform for learning Java programming (Hashim, A. 2007) and Microsoft TouhDevelop a programming environment intended to enable anyone to use a phone to program the phone using scripts for their windows based smartphones (Athreya, B. Et al. 2012).

Holz et al (2011) researched on integrating smartphones as interesting everyday objects into computer science courses to raise motivation and the results show that using smartphones can have a highly motivational effect on students of both sexes and the usage of the latest media and technologies is generally far more motivating than the usage of the old and long-known ones. Eighth grade students were taught programming using TouchDevelop the results are, 7% of the students stated that they thought the development of a mobile application with TouchDevelop was easy, 48% thought it was somewhat difficult, and 45% said it was difficult. However, 86% of the students wanted to create more applications using TouchDevelop. 92% of the girls wanted to continue writing applications whereas only 38% of them had an idea of what applications they could create prior to the class (Tillmann N. et al. 2012). This shows the potential of using smartphones as a tool for teaching programming. However, Alsaggaf (2012) points out that while most students today own and use mobile devices, these devices are not obviously utilized as practical learning aids in lectures. Furthermore, these tools also lack visualization which is very important in learning programming.

4. DYNAMIC PROGRAM VISUALIZATION ON ANDROID PHONES

Dynamic program visualization engine on android smartphones is designed to support basic common programming concepts written in Java programming language syntax. Dynamic visualization is provided for sequential instruction execution, variable declaration and assignment, Expression Evaluation, data input, message output, selection, loop and function calls.

4.1 System Architecture

The system contains the user interface, Tokenizer, Parser, Interpreter and visualization Engine as shown in figure 1. User interface includes Editor and Visualization Pane. User writes code using editor and source code is tokenized and parsed before interpretation.

Interpreter sends command to visualization engine which animates code execution on visualization pane (4,5), during code execution visualization user can enter input to the program (6) and line of code in execution is highlighted in the Editor (7).

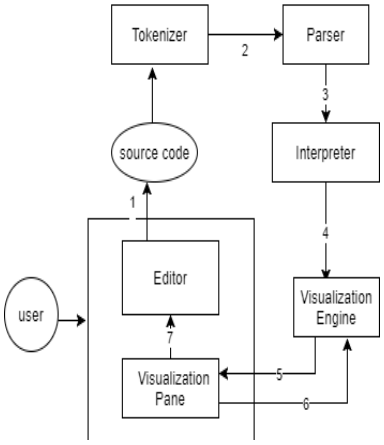

Figure 1. System Architecture

Figure 2. Source Code Editor

4.2 Program Visualization and Animation Control

The first stage in dynamic program visualization is to write source code in the source code Editor The editor contains buttons for compiling and controlling program visualization as shown in figure 2. Play button is used for compiling and starting program animations. It changes to stop button during program visualization so user can stop program visualization and go back to edit code. Forward and backward buttons are used for step wise animation. Forward and backward buttons are activated by pressing pause button.

During program execution, user is provided with graphical representation of program execution. Program in figure 2, includes sequential instructions, variable declaration and assignment, data input, message output and Expression Evaluation is used to demonstrate the dynamic program visualization. Sequence of instruction execution is animated by highlighting with pale dark blue color the instruction number of the current instruction (see instruction number 4 in figure 3).

Variable declaration is shown by adding a row in a table with first column presenting variable name and second column shows data stored in that variable. When data assignment instruction is executed, the corresponding data field in a variable table changes from null to the assigned data value. Instruction number 7 of the source code, the program asks input. In executing this instruction a EditText view pops up to the right side of variable declaration, where user can type in the value. The typed in value is assigned to the corresponding variable as is entered. Figure 4 above shows the presentation.

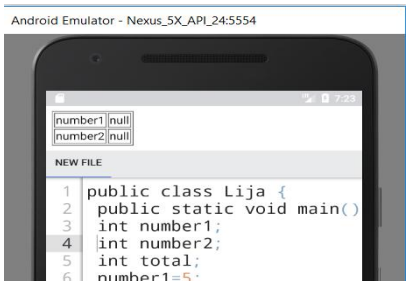

Figure 3. Variable Declaration

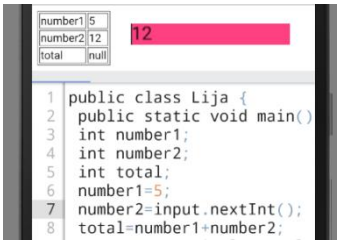

Figure 4. Data Input

Note that visualization designed does not support Object oriented programming concept. Hence input.nextInt() is not related to a method nextInt() in Scanner class of the java.util package but it is defined as the instruction to get input from the user so that user can input data during visualization. Same case applies to System.out.println().

Expression Evaluation is visualized by a pop up TextView which appears to the right of variables declaration. This region is used also for data input and output message. Instruction number 8 is expression evaluation, a TextView pops up to the right of the variable declaration visualization and shows the values of variables instead of variables names of the expression, the expression result is stored to the variable. Expression evaluation is shown in blue color to distinguish it from data input which is shown in red color and message output which is show in yellow.

Finally message output instruction number 9, message output is acting like a console in standard compilers and IDE. The message is shown with yellow background TextView to the right of the variable declaration.

5. CONCLUSION

Currently, program visualization for Selection, Loop and Functions are under development. Due to small screen size it is challenging to visualize classes and objects therefore the visualization of these programing concepts will not be provided. However it is expected that the tool will be useful for novice programming learners in learning basic programming concepts. It is also expected that the tools can enable a large number of students in resource challenged environments to learn computation skills as it can be accessed on widely available smartphones. The process of evaluating this tool in learning environment is under way.

REFERENCES

Alsaggaf, W., 2012. Enhancement of learning programming experience by novices using mobile learning: mobile learning in introductory programming lectures. *Proceedings of the ninth annual international conference on International computing education research*, ACM , pp. 151-152.

Athreya, B. et al, (2012). End-user programmers on the loose: A study of programming on the phone for the phone. *In Visual Languages and Human-Centric Computing (VL/HCC), 2012 IEEE Symposium*, pp. 75-82.

Bhattacharya, P. et al, 2011. A Collaborative Interactive Cyber-learning Platform for Anywhere Anytime Java Programming Learning. *In Advanced Learning Technologies (ICALT), 11th EEE International Conference*, pp. 14-16.

Čisar, S. et al, 2011. Effectiveness of program visualization in learning java: a case study with jeliot 3. *International Journal of Computers Communications & Control*, pp. 668-680.

Hashim, A., 2007. Mobile Technology for Learning Java Programming-Design and Implementation of a Programming Tool for VISCOS Mobile. *Unpublished Master's thesis*, University of Joensuu, Finland.

Holz, J. et al, 2011. Using smartphones to motivate secondary school students for informatics. *In Proceedings of the 11th Koli Calling International Conference on Computing Education Research* , ACM, pp. 89-94.

Jesse, G. R., 2015. Smartphone and App Usage Among College Students: Using Smartphones Effectively for Social and Educational Needs. *In Proceedings of the EDSIG Conference*, pp. 3424.

John, M. S. and Rani, M. S., 2015. Teaching Java Programming on Smartphone-pedagogy and Innovation; Proposal of its Ontology Oriented Implementation. *Procedia-Social and Behavioral Sciences*, *176*, 787-794.

Moreno, A. et al, 2005. Jeliot 3, an extensible tool for program visualization. *In 5th Annual Finnish/Baltic Sea Conference on Computer Science Education*.

Pan TY., 2017. Reenergizing CS0 in China. *In: Rich P., Hodges C. (eds) Emerging Research, Practice, and Policy on Computational Thinking. Educational Communications and Technology: Issues and Innovations*. Springer, Cham

Pears, A. and Rogalli, M., 2011. mJeliot: ICT support for interactive teaching of programming. *In Frontiers in Education Conference (FIE)*, IEEE, pp. T1J-1.

Smith M., 2016. Computer Science For All, https://obamawhitehouse.archives.gov/blog/2016/01/30/computer-science-all, accessed Nov 2017.

Tillmann, N. et al, (2012). The future of teaching programming is on mobile devices. *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education*, ACM, pp. 156-161.

United Nations Data., 2017. http://data.un.org , accessed December 2017