TOOL SUPPORT FOR LEARNING COMPUTER AND ROBOT PROGRAMMING

Obe O. Olumide[1] and Tiko Iyamu[2]
[1]*Federal University of Technology, Akure, Nigeria, Computer Science Department, School of Computing*
[2]*Cape Peninsula University of Technology, Cape Town, South Africa*

ABSTRACT

Computer Programming is believed to have effect on creativity, reasoning, analytic and mathematical skills. This cognitive development is at a cost from both ends of students and teachers of computer programming. Its abstractive nature makes it difficult to teach and learn hence, the enormous hours spent in teaching, learning and developing solutions. Consequently, the less interest among the students at all levels of education. Studies have shown various attempts to ease its teaching and learning by developing user friendly interfaces and use of course video teaching clips from seasoned programmers. However, learning and teaching programming skills remains a herculean task for student and teachers respectively. It is believed that interacting with a more tangible object can improve students' interest in developing computer programming skills. A mobile robot- a situated and embodied electronic device serves this purpose hence, the use of the device that students have more interest in interacting with in this study. In this work, an interactive program evaluation framework was developed that interact with a simulated e-puck robot on V-Rep simulator platform. The study shows an increased interest and improved performance in computer programming among Computer Science students.

KEYWORDS

Simulator, Programming, Mobile Robot, Learning, Python, Artificial Intelligence, V-Rep, Collaboration

1. INTRODUCTION

Computer programming becomes boring to new Computer programming students due to lengthy time spent in testing and debugging. It was estimated that software testing takes about 45% of time spent in software development (Obe, 2017). In order to sustain technological development that adopts more of software integration and encourage more participation in software development; methods, tools and techniques must be developed to simplify teaching, learning and development of computer software across all platforms among students at all levels of education. Siegfried et al. (2017) identified mobile robot as a veritable tool for teaching computer programming among school children because they are embodied and situated. These tangibility and sociability features of the mobile robots are necessary for assisting in conveying and impacting knowledge and skill sets.

Asides using mobile robots to assist teaching and learning computer programming, there exist needs to meet up with the minimum bench mark requirements in tertiary institutions for Computer Science students in Artificial Intelligence curriculum where robot programming is encouraged. A real-time program assessment framework for robot programming must be developed to assess the level of comprehension of various subject matters of the curriculum. Invariably, while the students are mastering computer programming fundamentals, they would as well be mastering the fundamentals of robot programming and the students can also implement more complex data structures for executing more complex tasks as necessitated by the robotic environments.

The use of mobile robot in teaching and learning especially, computer programming has been explored and researched on by many researchers using different tools, methods and techniques. Some of which were discussed in the next section. The research question is: what is the relevance of Robot programming application tool in teaching and learning Computer programming?

2. RELATED RESEARCH

The research findings of Pugnali et al., (2017) suggest that type of user interface does have an impact on children's learning, but is only one of many factors that affect positive academic and socio-emotional experiences. Different kinds of learning qualities are obtainable through graphical interfaces and tangible devices like robot.

Esteve-Mon et al., 2019 established the effectiveness of adoption of educational robotics in the growth of computational thinking of students, especially among male students. Mather, (2015) explored a mixed-method approach for evaluating technology-enhanced learning environments for Computer programming in early stage students. The finding shows an enhanced productive collaborative behavior and improved learning among learners by creating a learning environment and use of 3D animated Ceebot. *Nourbakhsh et al.* (2004) explicitly evaluated the educational impact of robotics on secondary level students by presenting statistically significant evidence of broad learning. Robert (*Róbert et al.*, 2010) designed method to help students acquire the essential programming structures visually with the help of LEGO NXT robots. In Major *et al.* (2011), the research was directed towards effort to help teach novices programming using physical robots. The effectiveness of using robots as tools in the teaching of introductory programming was investigated, and the impact of the technology in helping to overcome the current barriers for leaners was assessed. *Ernest et al* (2011) researched on the effectiveness of integrating Educational Robotic Activities into higher education Computer Science Curricula: A Case Study in a Developing Country. Marina et al, (2013) worked on: Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. This research was motivated by the needs to expand robot programming in early student curriculum. CHERP (Creative Hybrid Environment for Robotic Programming) was developed that enabled children to dedicate their time building robot, planning its actions, using a physical wooden block or the computer screen to construct programs, and iteratively improving the robot and programs according to initial goals and subsequent discoveries. Attila et al. (2013) developed a tool to make the learning process of programming languages more concrete, practical and interesting for students, with an optimal environment and strongly limited set of statements. Park et al. (2015) developed systematic and effective robot-based learning with programming to improve learner's creativity and understand class satisfaction at the elementary school level in Korea.

Carina et al. (2016) worked on enhancing the Engagement of Intelligent Tutorial Systems through Personalization of Gamification. Tocháčeka et al. (2016) worked on identifying the potential and aspects of exploitation of educational robotics project in education at secondary schools and training teachers. Simon et al. (2017) in 'An exploration of the role of visual programming tools in the development of young children's computational thinking' explored the impact of young children's programming approaches to programming in two tools with contrasting programming interfaces, ScratchJr and Lightbot on developing computational thinking.

In this work, a program assessment framework for robot programming based on running navigation simulations in a known environment through a real-time implementation which provides score feedback and gamified hints to the students was developed.

3. RESULTS AND DISCUSSION

3.1 Research Method

This section presents the system architecture showing the interactions between the main components of the system.

The system trains a new user on robotics programming through detailed documentation and analysis on each user input. It achieves this by providing the user with tasks related to areas in robotics e.g. actuation, computer vision, real time logic decision making etc and analyzing each user input in real time. The user's performance is graded and the resulted is displayed at the end of each task. The system consists of five levels each consisting of four tasks or more.

3.1.1 System Architecture

The robot programming assessment system is composed of a test library, a simulator, and the user interface as depicted in figure 1. The python test library provides the user with the set of task to be executed, with each task containing a script describing the actions to perform during simulation.

The V-rep simulator loads the scenes that would be presented on the user's interface, which is connected together by the python remote API.

The GUI displays scenes loaded from the V-rep simulator, displays tasks to be executed at each level/stage within a specified time, then, it output scores feedbacks to the user at the end of each stage.

Implemented in each stage are set of tasks to be completed by a player. Therefore, the overall assessment is computed based on the feedback score gotten at the end of each stage.

This section presents the system architecture (Figure 1) showing the interactions between the main components of the system. The system is a desktop application written in the python programming language leveraging the framework for its graphical user interface (GUI).

The system was designed to interface with an existing simulation environment called VREP. The system starts up VREP in a server mode with the address '127.0.0.1' and establishes a connection as a client on the port '19998' as depicted in figure 2.

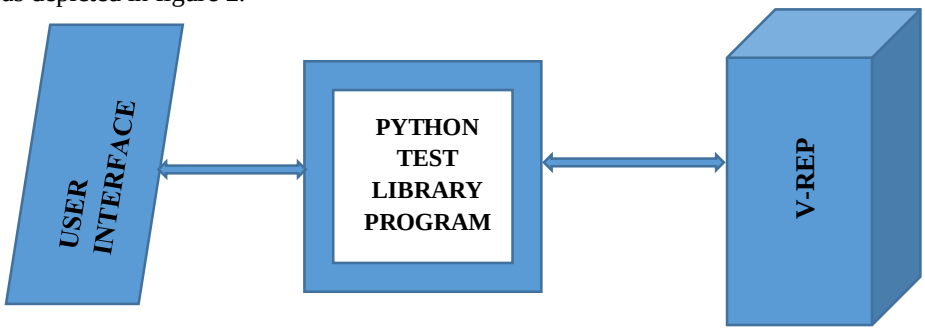

Figure 1. System Architecture of the Application

3.1.2 How the Application Interacts with the User

Figure 2 depicts how the desktop application will interact with the users of the system and the overall behavior of the system.

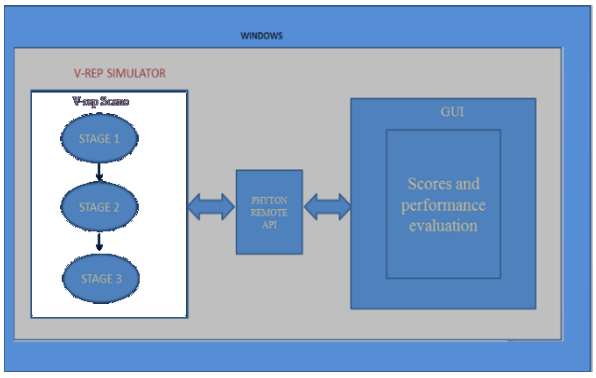

Figure 2. Breakdown of the System Architecture of the Application

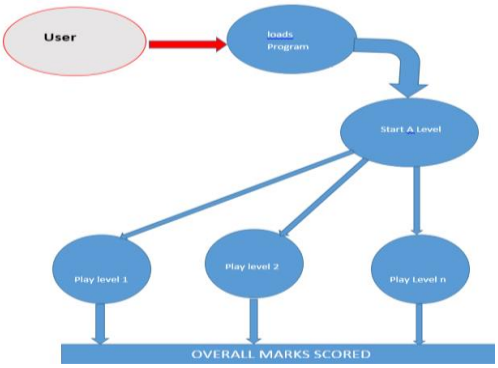

Figure 3. Application Interaction flow with the User

The robot program evaluation system was implemented with the E-puck robot in a V-REP (Virtual Robotic Experimentation Platform) simulator programming environment embedded with Lua scripts by defaults. V-REP is a comprehensive CAD-based application that can be used to simulate, test and program a robot from a 3D CAD environment, allowing users with basic robot programming skills to generate robot programs off-line.

The assessment system is composed of a test library, simulator, and the graphical user interface. The test library is built using a python programming language to provide the set of task to be executed, with each task containing a script describing the actions to perform during simulation. While the simulator contains scenes formulated on the V-REP simulator.

The remote API used for this work is extended by custom Lua functions which are recognized from their simx*, -prefix. The remote API is part of the V-REP API framework which supports remotely loading a scene,

starting, pausing or stopping a simulation. The remote API functions are interacting with V-REP via socket communication (or, optionally, via shared memory) in a way that reduces lag and network load to a great extent. All this happens in a hidden fashion to the user.

The Graphical user interface (Game Play) is designed with PyQt, to implement the play button, Points gotten at each stage, the overall points, and countdown time. Then, python remote API functions was used to load the scenes from the V-REP simulator, corresponding to a particular stage of the game. When the time required to complete a task finishes, the script automatically rank the player and loads the next stage of the game. This repeats until the game is over.

Building on the evaluation framework, several hints for a task in which the students must program a robot (using control flow instructions) to navigate in a known environment within a specified time is given. Therefore, the assessment is based on the feedback score and hint gotten at the end of each test scenarios.

To test the program in a different environment of increasing complexity, the task is split into 3 stages, each containing navigating round an empty space, navigating from a certain point to the goal, and path following respectively.

3.1.3 Programming Language

There exists hundreds of programming languages but very few of them are applicable in programming robot. Python programming language, C/C++ and Java are the best languages for robot programming. Technically, C & C++ **are most appropriate start point for a new** roboticists, **because they contain a lot** of the hardware libraries which **allows** low-level hardware interaction resulting in time performance. They present very rich data structures and helps in learning other programming languages (i.e Java, Python, etc) in their categories. This work attempt teaching C/C++ languages using simulated e-puck robot on V-Rep simulator.

3.1.4 Task and Level

Robots are programmed to carry out tasks. Each task carries out a well-defined operation that impact the robot environment using its actuators. To carry out a mission, there may be need to carry out a number of tasks. There are low level task with low computational complexities. Complexity of the task dictates the kind of data structures, control structures, library functions/methods and algorithm to be implemented and invariably affect the mission. In a mission, a robot may have to switch several times between a numbers of tasks depending on the context as presented by the environment. Having the robot to successfully and efficiently perform depends on the kind on optimization algorithm implemented to optimize the robot movement and to perform the desired tasks. These tasks and algorithms implement data structures and control structures which are fundamental in learning computer programming. In this work, we classify these structures according to some level of difficulties and stages/lessons to be learned by the students (in their various curricula) and integrate them into tasks and levels.

Tasks Categories: in this work, we only consider elementary tasks for educational mobile robot which includes the following: wandering; moving on a straight line; drawing of shapes; following a line; following an object; obstacles avoidance; moving to a target position; follow leader; cooperative behavior; and swarm intelligence etc.

3.1.5 Tasks Levels

The tasks are classified as shown in Table 1. There are several number of tasks questions/problems formulated using problem solving and critical thinking approaches (Morin et al., 2015) under same level. Questions of the same difficulty level are randomly generated under the same level while difficulty increases across levels.

Table 1. Task Classification

Level 1 tasks	Level 2 tasks	Level 3 tasks
Wandering in free space Using simple commands	Drawing of shapes (circle, triangles etc.)	Target reaching
Line following (straight, curves, loops etc.)	Object following	Leader following
Obstacles avoidance		Implementing swarm behaviour algorithms

3.1.6 Mathematical Representation of the Assessment System

At each level, a number of tasks are assigned. Six tasks are randomly assigned at each level except for the first level where only four tasks are assigned. For each successful tasks assignment completion, a corresponding score is assigned. The accumulated score from each tasks completed successfully over a number of levels are summed up using the following mathematical expressions:

Level 1 (score) :

$$\sum_{i=1}^{n=4} X_i\,(t)$$

…

Level n (score) =

$$\sum_{i=1}^{n=6} X_i\,(t)$$

where;

X_i=marks for each task

n=number of tasks in each level

NB: Countdown time (t) is constant

Total Marks= ∑ ((level 1) + (level 2) + … (level n))

3.2 Results

How the Application Starts: The user launches the application and the system displays a screen with a loader and loads important modules into memory as shown in figure 4.

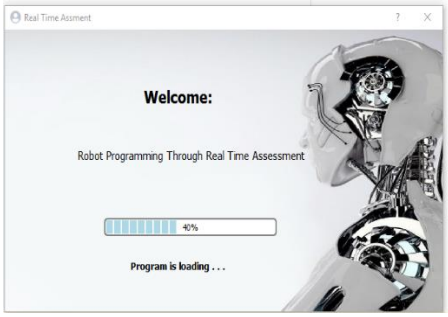

Figure 4. System Loading

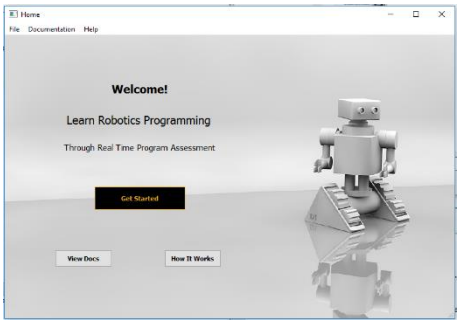

Figure 5. Home Screen

The system finish loading modules and displays home screen (see figure 5). The user then clicks the start button to begin play. On click, the python remote helps to connect the GUI with the V-Rep simulator, then, several scenes will be loaded from the VREP simulation environment.

The game has 3 stages. In each stage, the player is expected to complete a task of programming e-puck robot with Python scripts inside V-Rep simulator to perform a single robotic control task.

How the Game Begin: User clicks start button on the home screen and the system loads screen for level 1 where the user starts level 1 by clicking on “level 1” (figure 6).

On Click, the system loads screen for level 1 where the user starts level 1 by clicking on “level” and a countdown timer starts

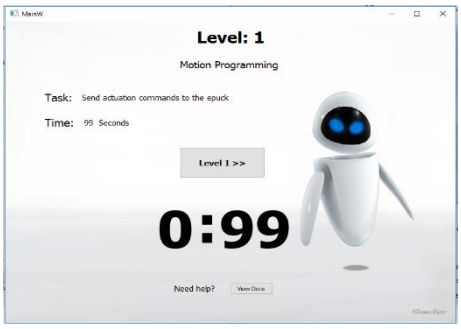

Figure 6. Level 1 GUI Screen with countdown timer

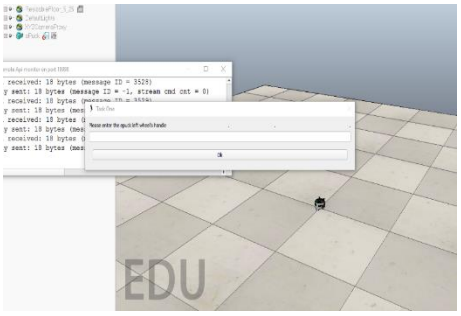

Figure 7. Level 1 V-Rep code Input Screen

The system then displays tasks for level 1 and accepts code inputs from the user as depicted in figure 7. The system displays success message if input is correct and error otherwise (figure 8).

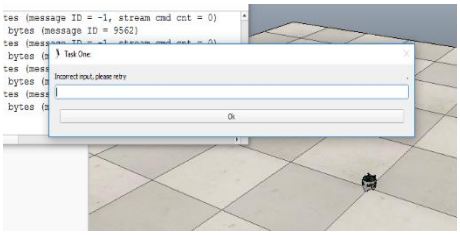

Figure 8. Wrong input entered

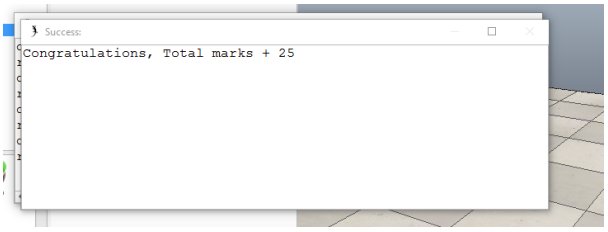

Figure 9. Right Input Entered

Once the user input the right code, a congratulation messgae will be displayed and the required score(marks) will be awarded for the level completion as shown un figure 9.

This process continues until the three levels are completed and the overall score is displayed. Therefore, at the end of a level, the tasks done will be combined together, and the simulation will be displayed for the user to see work progress. The user may also decide to take test only at any of the levels depending on the area of the curriculum of the programming language covered. Figures 10a and 10b depicts some of the results at level 1.

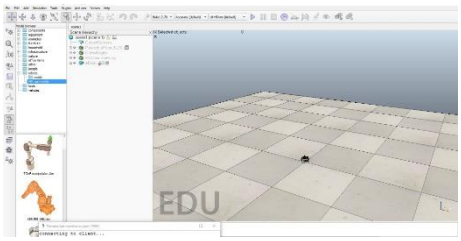

Figure 10a. Simulation of level 1- free wandering

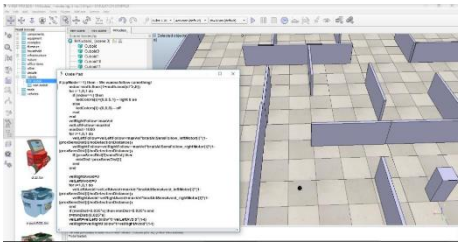

Figure 10b. Simulation of level 1- Obstacle Avoidance and wall following

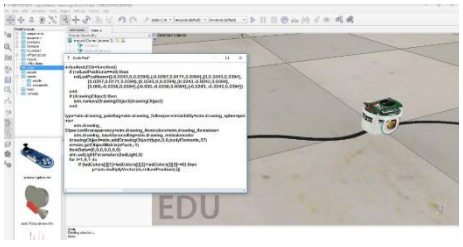

Figure 10c. Simulation of level 1- Line following

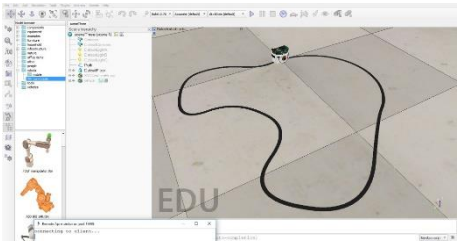

Figure 11. Simulation of Level 2-Shape drawing

Figure 11 shows one of the results at the level 2 of the training. The robot was programmed to draw shapes.

Figures 12a and 12b shows the result at the Level 3. The robot was programmed to move from a starting point to a target point (figure 12b) in the simulated robotic environment.

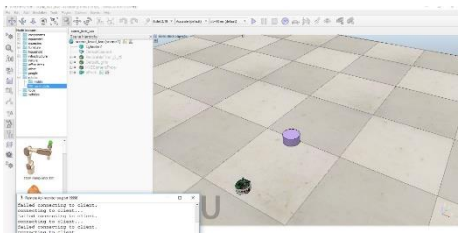

Figure 12a. Simulation of Level 3-Target Reaching

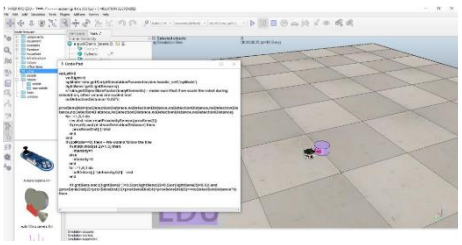

Figure 12b. Simulation of Level 3-Target Reaching

3.2.1 User Documentation

To aid user learnability, the game is embedded with a documentation which explain how each level works, how to learn basic python codes for Robot programming, and how to successfully work around various V-REP scene.

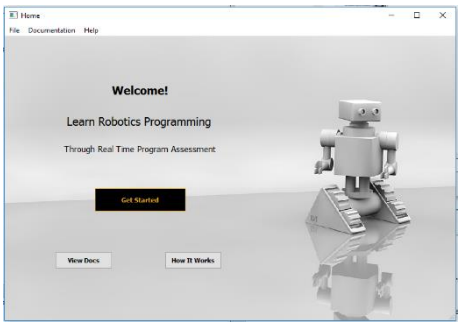

Figure 13. Documentation Button

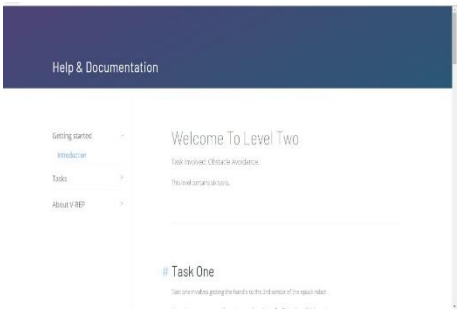

Figure 14. Documentation Window

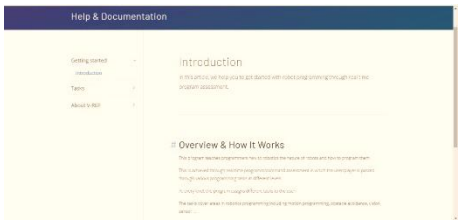

Figure 15. How the Game System Works Documentation

4. CONCLUSION

This work has presented an assistive system to aid novice computer and robot programming students to learn how to program and at the same time learn how to program a robot. It demonstrated a gamified hint system based on the assessment of robot programs. The system is able to evaluate students' programming skills in real time. By running the program inside the simulator, several scenarios were tested using the basic robot control functions.

However, by providing tutorials and hints to the students, they are eager to learn how to program and the awarding of grades at the end gives them the urge to take robot programming serious because every student would want to put in his best in order to score great points in the end.

5. CONTRIBUTION TO KNOWLEDGE

This work has developed an educative tool for teaching and learning computer programming through the use of robot programming. The real-time programming assessment tool will serve as an effective teaching aids to improve teaching and learning process of computer and robot programming.

REFERENCES

Attila P., Robert P., & Erika T. (2013) 'Mobile Robots in Teaching Programming for IT Engineers and Its Effects' International Journal of Advanced Computer Science and Applications, Vol. 4, issue 11, pp. 162-168 Doi: 10.14569/IJACSA.2013.041123

Carina S., Pedro T., & Vanesa M. (2016) 'Enhancing the Engagement of Intelligent Tutorial Systems through Personalization of Gamification' International Journal of Engineering Education Vol. 32, issue 1(B), pp. 532–541

Ernest B., Hanheide M., & Cielniak G. (2011) 'The Effectiveness of Integrating Educational Robotic Activities into Higher Education Computer Science Curricula: A Case Study in a Developing Country' A conference proceeding from the book 29 Effective Ways You Can Use Robots in the Classroom. (pp. 73-87) DOI: 10.1007/978-3-319-55553-9_6

Esteve-Mon, F. M., Adell-Segura, J., Llopis Nebot, M. A., Valdeolivas Novella, G., & Pacheco Aparicio, J. (2019). The development of computational thinking in student teachers through an intervention with educational robotics. Journal of Information Technology Education: Innovations in Practice, 18, 139-152. https://doi.org/10.28945/4442

Major L., Kyriacou T., & Brereton O. (2011) 'Systematic literature review: Teaching Novices Programming using Robots' School of Computing and Mathematics, Keele University, Staffordshire, UK Vol. 6, Issue 6, pp. 502 – 513. Doi: 10.1049/iet-sen.2011.0125

Marina U., Louise F., Elizabeth R., & Amanda S. (2013) 'Computational thinking and tinkering: Exploration of an early childhood robotics curriculum'. A journal of Computer & Education Vol. 72, pp. 145-157. https://doi.org/10.1016/j.compedu.2013.10.020

Mather, R. (2015). A mixed-methods exploration of an environment for learning computer programming. *Research in Learning Technology*, *23*. https://doi.org/10.3402/rlt.v23.27179

Morin, D., Thomas, J. D. E., & Saadé, R. G. (2015). Fostering problem-solving in a virtual environment. Journal of Information Technology Education: Research, 14, 339-362. https://doi.org/10.28945/2273

Nourbakhsh I, Hamner E., Crowley K., & Wilkinson K. (2012) 'Formal measures of learning in a secondary school mobile robotics course' A Conference Proceedings in the IEEE International Conference on Robotics and Automation Vol. 2: pp. 1831 - 1836 DOI: 10.1109/ROBOT.2004.1308090

Obe (2017). Test Data Generation Using Intelligent Geno-Neuro Technique. International Journal of Scientific & Engineering Research Volume 8, Issue 8, 650-655.

Park I., Kim D., Oh J., Jang Y., & Lim K. (2015) '**L**earning Effects of Pedagogical Robots with Programming in Elementary School Environments in Korea' Indian Journal of Science and Technology. Vol. 8, Issue 26. DOI:10.17485/ijst/2015/v8i26/80723

Pugnali, A., Sullivan, A., & Bers, M. U. (2017). The impact of user interface on young children's computational thinking. Journal of Information Technology Education: Innovations in Practice, 16, 171-193. Retrieved from http://www.informingscience.org/Publications/3768

Siegfried R., Klinger S., Gross M., Sumner R. W., Mondada F., & Magnenat S. (2017) Improved Mobile Robot Programming Performance through Real-time Program Assessment. In ITiCSE '17: Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education, pp. 341-346 https://doi.org/10.1145/3059009.3059044

Simon P., Jacob P., & Tim J. (2017) 'An exploration of the role of visual programming tools in the development of young children's computational thinking' An Electronic Journal of e-Learning Vol.15, Issue 4, pp. 297-309.

Tocháčeka D., & Lapesa V. (2016) 'Developing technological knowledge and programming skills of secondary school students through the educational robotics projects' A conference proceeding on Social and Behavioral Sciences. Vol. 217, pp. 377 – 381. Doi: 10.1016/j.sbspro.2016.02.107