CJLT RCAT
Canadian Journal of Learning and Technology
La Revue canadienne de l'apprentissage et de la technologie

Volume 46 (1) Winter/Hiver 2020

# Application of Learning Objects for Computer Programming-Based Problem Solving

# Application des objets d'apprentissage à la résolution de problèmes basée sur la programmation informatique

***Abdullah Ahmad Basuhail,*** *King Abdulaziz University*

**Abstract**

This paper presents an approach to implement learning objects for teaching and learning problem-solving techniques based on computer programming. The demonstrated approach exploits computer-based interactive animations and computer graphics. The main feature of this approach is its simplicity for exploring the concepts and structures of the programming that are used to implement a solution for a problem under consideration. The developed learning objects feature the possibility of reusability and adaptability in e-learning settings. Moreover, the learning objects can be utilized as a hands-on experience for the learners of a certain subject matter. The approach applied for the design and implementation of the learning objects for computer programming-based problem solving can be extended to other disciplines of science and technology. As a demonstration of the proposed methodology, we showed an application that utilizes the approach to implement a learning object for solving a well-known statistics and probability problem.

*Keywords:* programming-based problem-solving, learning objects, computer animation, programming, e-learning, educational technology

**Résumé**

Ce document présente une approche pour la mise en œuvre d'objets d'apprentissage pour l'enseignement et l'apprentissage de techniques de résolution de problèmes basées sur la programmation informatique. L'approche démontrée exploite des animations interactives et des graphiques sur ordinateur. La principale caractéristique de cette approche est sa simplicité pour explorer les concepts et les structures de la programmation qui sont utilisés pour mettre en œuvre une solution à un problème considéré. Les objets d'apprentissage développés présentent la possibilité de réutilisation et d'adaptation dans des contextes d'apprentissage en ligne. De plus, les objets d'apprentissage peuvent être utilisés comme une expérience pratique pour les apprenants d'un certain sujet. L'approche appliquée pour la conception et la mise en œuvre des objets d'apprentissage pour la résolution de problèmes basée sur la programmation informatique peut être étendue à d'autres disciplines scientifiques et technologiques. En guise de démonstration de la méthodologie proposée, nous avons présenté une application qui utilise l'approche pour mettre en œuvre un objet d'apprentissage pour résoudre un problème bien connu de statistiques et de probabilité.

*Mots clés* **:** résolution de problèmes basée sur la programmation, objets d'apprentissage, animation par ordinateur, programmation, apprentissage en ligne, technologie éducative.

## Introduction

Technology-based learning is vital in higher education. It is increasingly ubiquitous, specifically in the e-learning environment. This technology engagement in education assists learners to realize information and concepts effectively and promptly. Further, it could save valuable in-class time that could be devoted to more exploration of the topic. Technology-based learning attracts students' attention during the learning process. The exploitation of various technological tools in learning and teaching can help learners get the best possible education and prompt feedback (Lillejord, Borte, Nesje, & Ruud, 2018). For instance, different technology tools can support different learning styles, such as auditory, visual, and kinesthetic (Anderson, 2016; Awla, 2014; Ford, Robinson, & Wise, 2016; Hughes, 2016; Maseleno, Hardaker, Sabani, & Suhaili, n.d.).

Individual learners differ in their cognitive skills and their attitudes to learning (Krassimir & Lu, 2017). Recent studies have reported approximately 95% of instructors with diverse backgrounds concluded that students learn better if instructed in their favoured learning style (Willingham, Hughes, & Dobolyi, 2015). On the other hand, the theory of learning styles has been criticized. Many educators believe there is no evidence that sufficiently confirms an association between learners' favoured learning styles and the corresponding instructor's teaching style (Pashler, McDaniel, & Rohrer, 2008).

Learning modules enriched with computer-based learning elements, approaches, and techniques, such as computer animations and graphics, can properly fulfill the needs of most or all learning styles. As a result, the gap in compatibility between instructors' teaching styles and learners' learning styles becomes eliminated or minimized.

Learning objects supported with animations and graphics are effective and powerful tools used in various disciplines of science and technology for introducing instructional concepts to the learners. They offer active methods to show the temporal change in a certain instructional topic. They are used to help instructors to explain instructional concepts.

Models for instructional design that fit specific contexts and learners become increasingly important. In those models, computer animations and graphics are exploited as part of the learning process. Computer-based technologies, such as animations and graphical elements in learning, can help learners intuitively grasp the instructional information involved in a certain topic from the demonstration implemented for this purpose.

This paper illustrates an approach that exploits learning objects supported by animations and graphics to teach and learn programming concepts used for problem solving. It illustrates a method for tracking program execution mechanism related to problem solving. This paper demonstrates an application of this methodology to implement a learning object for a well-known problem.

The learning objects can be designed and implemented for the teaching and learning of various computer programming concepts to learners from different disciplines.

## Learning Objects

A learning object is considered as information and/or practice entities that are all integrated to reinforce a single educational goal. It plays a significant role in the applied experience of the learners.

Learning objects are described by many definitions. Wiley (2000) describes learning objects as elements of a type of computer-based instruction, which can be reused in different learning contexts and are generally understood to be digital entities. The Institute of Electrical and Electronics Engineers (IEEE) defines a learning object as "any entity, digital or non-digital, that may be used for learning, education or training" (IEEE, 2002). The National Ministry of Education of Colombia defines learning objects as “any digital resource that can be reused to support learning” (Wiley, 2000). A learning object is defined as a digital, self-contained, reusable entity with a clear learning objective that contains at least three internal varying components: content, instructional activities, and context elements. It should also have an external component of information, which helps its description, storage, and retrieval: the metadata (Chiappe, Cifuentes, & Rodríguez, 2007).

Adopting learning objects in teaching and learning gives the learners the chance for further exploration of the instructional concepts on their own. The use of dynamic learning objects instead of static learning elements throughout the learning process eliminates the need for supplementary explanations and the use of marking signs, so that illustrations can be simpler, clearer, and less cluttered. Moreover, the use of learning objects can support learning that is richer, attractive, and intuitive.

Learning objects offer education continuous improvements. They tend to lead to quality for higher education academic contents (Chiappe, Cifuentes, & Rodríguez, 2007). One of the reasons for the education quality and enhancements is that the learning objects should be stored in electronic online repositories that allow their public use. In general, those contents are subjected to continuous revision and critique from the higher education community (Chiappe, Cifuentes, & Rodríguez, 2007). This public use allows numerous and global evaluations not only by academic peers, but also by learners and practitioners, who score and make comments about the material by means of the available tools in those online repositories (Chiappe, Cifuentes, & Rodríguez, 2007). Later, those comments will be analyzed by the developers to make the required enhancements and updates. This kind of feedback diversity, analysis, and revision leads to improved educational quality.

Learning objects are applicable not only as a learning material, but also as a teaching strategy (Chiappe, Cifuentes, & Rodríguez, 2007). The mechanism of problem solving using computer programming-based solutions is usually illustrated by procedures, pseudocode, sequences, algorithms, flowcharts, and processes. In this paper, we suggest implementation of programming-based problem solving using digital learning objects.

**Programming-Based Problem Solving**

Problem solving is considered as the backbone of the curriculum of computer science and engineering for undergraduates. Basically, computer programs are written for problem solving. Many techniques, structures, and concepts are considered in the programming courses to aid in providing solutions to those problems. The programming structures and concepts include the use of numbers, constants, variables, initialization, manipulation, assignments, conditions, loops, arrays, algorithms, data structures, etc. Instructors exert efforts to introduce and describe the programming concepts and structures in ways that will help students to grasp them. They use illustrations, such as explanatory labels, textboxes, arrows, diagrams, and flowcharts, to explain the logic and trace the sequence of execution of the programs. However, these tools are static in their mechanism and they need further support and explanations to help students fully realize the approach used for problem solving. Furthermore, using these solutions in e-learning, self-learning environments need further support to be provided for the learners to grasp and fully understand the concepts and structures and afterwards craft a solution for the problem under consideration.

The use of learning objects can help and enhance the learning of the programming concepts and structures for problem solving. Their use can support and enrich the learning process. Moreover,

using this approach in the learning sessions can save significant time, which can be used for additional related practice and experience.

## Litreture Review

The use of computer animations and graphics in higher education courses has been addressed by many researchers. Some researchers used animations in teaching computing concepts (Smith & Escott, 2004). Others suggested animated models for teaching aspects of computer systems organization (Henderson, 1994). Researchers have analyzed the effectiveness of combining algorithm and program animation with data structure in courses (Osman & Elmusharaf, 2014). Some researchers directed analysis for using instructional animation, and reported significant benefit over the use of static elements (Hoffler & Leutner, 2007). Others studied animations and indicated their usefulness in simplifying learning (Hwang, Tam, Lam, & Lam, 2012). Some researchers observed improvement in students' achievements (Smith & Escott, 2004). Other instructors showed that animation provided supportive and entertaining tools to most of the learners (Lowe, 2004). Some researchers stated that using animations gives an exact and rich representation for some topics that are very hard to comprehend from text-based information (Falvo, 2008; Rotbain, Marbach, & Stavy, 2008). Several researchers concluded that the use of learning objects contributes significantly to the teaching-learning process of programming (Adamchik & Gunawardena, 2003; Begosso, Begosso, L., Begosso, Ribeiro, & Martins, 2015; Narasimhamurthy & Shawkani, 2009). Halverson, Wolfenstein, Williams, and Rockman (2009) described how the design of digital learning objects can spark professional learning.

The use of learning objects in teaching and learning programming specifically, and its effectiveness and challenges, has been addressed in many research studies (Narasimhamurthy & Shawkani, 2009). The interactive learning objects are appreciated by many instructors in search of new methods and supports for novice programming students (Matthiasdottir, 2006). Several studies showed that the use of learning objects contributes significantly to the improvement of computer science teaching-learning process (Adamchik & Gunawardena, 2003; Andreza & Magalhães, 2019; Begosso, Begosso, & Begosso, 2016; Begosso et al., 2015; Cavus & Ibrahim, 2004; Jaimez-González, García-Mendoza, Luna-Ramírez, Nápoles-Duarte, & Vargas-Vargas, 2018; Luna-Ramírez & Jaimez-González, 2014; Narasimhamurthy & Shawkani, 2009; Tapoli & Mikropoulos, 2019; Villalobos & Jiménez, 2009).

The distinction of our approach in this paper is in terms of providing a learning object design and implementation for the integration of programming-based concepts and structures with problem-solving techniques, rather than focusing solely on the programming structures.

## Methodology

The design and implementation of a learning object to illustrate the concepts and structures of the program codes written for problem solving can be realized by a procedural approach. Figure 1 shows a block diagram of the proposed stages of this approach. In the following sections these stages will be highlighted.

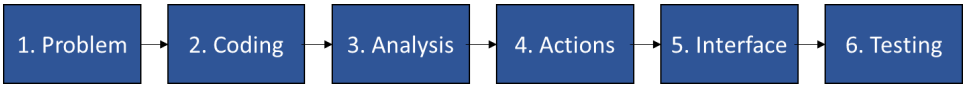


*Figure 1.* Learning object design model for computer-based problem solving.

In the first stage, the problem is well-described to provide a solution that will be implemented by a computer-based programming structure.

The second stage deals with coding, where a program script that provides a solution to the problem is coded.

Next, the code is divided into segments in stage three. Each individual segment is analyzed and partitioned into a sequence of structured components. The components are transformed into actions in stage four. An analysis table is generated to describe the program code by a sequence of steps.

In stage five, an interface of the learning object is designed. Graphical user interface components, such as textboxes, labels, and images, are added to the learning object. Those components will support the interpretation of the problem-solving concepts and structures by the learners. Moreover, control elements such as command buttons, radio buttons, checkboxes are incorporated to allow interaction and steering of the learning object in accordance with the logical sequence for the solution of the problem. Afterwards, the actions are imitated with the support of computer graphics and animations. The flow of the actions through the entire learning object is the most critical phase. Animation cascading is done by applying transitions steered by the problem under consideration. Animations are applied following the mechanism of the problem solving.

Often, the implemented problem-solving learning object is integrated for use in a teaching or learning environment, or it might be reused for functioning with other learning objects. The proper order of operation is considered to overcome any confusion that may arise from the outcome. A thorough testing procedure (stage six) is applied to ensure the exact and precise operation of the overall implemented learning object. In case of any operational inconsistency, the implementation stages are revisited.

**Application**

In this section, we will illustrate the application of the methodology to implement a learning object for programming-based problem solving, by the means of a scientific problem. The example considered here is a well-known problem in probability and statistics, which is rolling a six-sided dice for several times and examining the integer value of each roll that appears on the top face of the dice. This problem is of a random nature. A dice's face can show one possible integer value that falls in the inclusive range from 1 to 6. All the dice rolling needed for the experiment is completed, and the frequency of occurrences of each dice's face is calculated. Figure 2 shows the six possible outcomes of dice rolling.

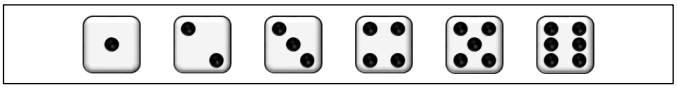

*Figure 2.* Six possible faces of dice rolling.

Here, the dice rolling problem is simulated using computer programming. The programming code simulates the experiment of rolling dice for 10 times, and the number that occurs on the dice's top face is checked and recorded. Figure 3 lists a code-segment written in the C# high-level programming language implemented for this purpose. Six counters are used to tally the occurrences of each dice's face. The corresponding counter is incremented according to the rolling outcome.

```
1. Random randNum = new Random();
2. int[] frequency = new int[7];
3. int face, roll;
4. for (roll = 1; roll <= 10; roll++) {
5. face = randNum.Next(1, 7);
6. frequency [ face ]++; }
```

*Figure 3.* Code-segment for the problem of rolling a dice.

The code-segment comprises several programming structures and concepts. Specifically, the code includes random number declaration and generation, array creation and initialization, variables declaration, loop, iteration, assignment operations, random number range, array manipulation, index entry, assignments, and increments. A one-dimensional array is used to tally the number of times each possible face appears. Based on the dice rolling problem, Table 1 lists the used concepts and structures and their association with the programming code.

Table 1

*Programming Concepts and Structures in the Code of Dice Rolling Problem Solving*

| Line/Code | Programming Concept/Structure |
|---|---|
| 1 | randomization declaration |
| 2 | array creation and initialization |
| 3 | variables declaration |
| 4 | loop, control variable, iteration, initial value, stopping condition, increment |
| 5 | random number generation, generated range, assignment operation |
| 6 | array-index entry, array manipulation, increment |

Table 2 lists the analysis steps of the code for the actions to be performed in the learning object based on the structures and concepts intended to be learned and grasped from the programming code of the problem solving.

Table 2

*Analysis of the Program Code*

| Program Code | Concept/Structure Description |
|---|---|
| frequency = new int[7] | 1. use array of 7 elements<br>2. initialize all the array elements<br>3. index 0 is never used |
| for (roll=1; roll<=10; ++roll) | 1. iterate for 10 times<br>2. use *roll* for loop variable<br>3. initialize *roll* by 1<br>4. increment roll variable<br>5. check stopping condition |
| face = randNum.Next(1,7) | 1. use *face* variable<br>2. generate random number<br>3. assign random to *face* |
| frequency[face]++ | 1. use *face* value as an entry to the *frequency* counters<br>2. increment corresponding frequency counter by 1 |

Figure 4 demonstrates a display that crafts an interface for the learning object for the dice rolling problem. The exhibit comprises seven components: 1) one box that represents the variable *roll*, which will display the loop iteration; 2) *face* that corresponds to the dice to be rolled; 3) seven boxes that represent the *frequency* counters, each of which will hold the tally of the occurrences of the corresponding dice face; 4) *table* on which the rolled dice will settle; 5) *Roll* control button for steering actions of the learning object; 6) *textbox* for a brief textual description of the problem; and 7) *code* box to facilitate mapping between the lines of code and the performed actions in the visualization of the learning object.

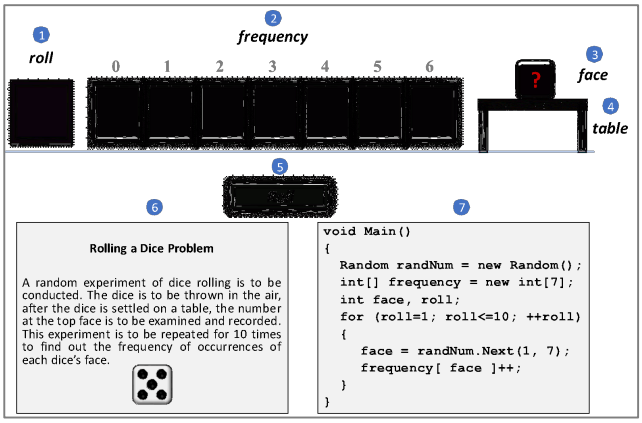


*Figure 4.* Exhibit for the learning object of rolling a dice problem solving.

Next, based on the nature of the problem under consideration and the analysis of the programming code, the actions are generated to be applied in the learning object to show the mechanism of the programming-based problem solving. For this specific problem, the generated actions are listed in Table 3.

Table 3

*Actions Sequence for the Dice Rolling Learning Object Problem Solving*

| Actions Generation |
|---|
| 1. display the learning object initial state |
| 2. wait until *Roll* control button is pressed |
| 3. disable *Roll* control button |
| 4. display 0s in all *frequency* counters |
| 5. disable *frequency* counter number 0 (dim) |
| 6. enable *Roll* control button |
| 7. wait until *Roll* control button is pressed |
| 8. display 1 in *roll* box |
| 9. repeat steps: 10-18 |
| 10. disable *Roll* control button |
| 11. throw the dice randomly upward and settle it on the *table* |
| 12. blink and change color of *frequency* counter number matching *face* |
| 13. increase value of matching *frequency* counter by 1 |
| 14. reset color in 12 back to original |
| 15. enable *Roll* control button |
| 16. wait until *Roll* control button is pressed |
| 17. increment *roll* box by 1 |
| 18. until *roll* box value exceeds *n* |
| 19. change *roll* counter box value color (to indicate termination) |
| 20. Display *Done* message |

## Discussion

A complete learning object for the problem solving of the dice rolling was implemented using the tools of animation software. The implemented learning object can be used in the learning of programming structures and concepts of problem solving. The learner can start using this learning object by pressing the *Roll* control button. This will enable the execution of the learning object.

Figure 5 displays the learning object's initial state. It demonstrates a frame that shows the concept of initialization of the frequency counters, which corresponds to the array used in the programming code. All the counters have been initialized with the value 0. Moreover, the frame illustrates the possible range of the outcomes of the random probability experiments, as face 0 can never appear from any dice roll. This is indicated by the dimming of the counter with entry 0 among the set of the frequency counters.

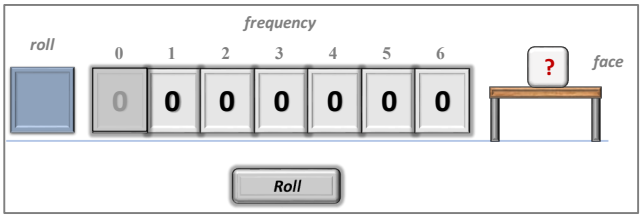


*Figure 5.* Learning object's initial state.

Figure 6 demonstrates two frames from running the learning object after pressing the command button *Roll* for one time. The first frame illustrates the concept of random number generation, where the dice is rolled upward, falls downward on one of its six faces, and then settles on the table. The second frame shows the situation after the completion of this rolling trial in this programming code loop iteration, where the dice settled on its 2-face. The frequency counter inside box 2 is incremented by a value of 1, as a result of the observed outcome from this roll trial.

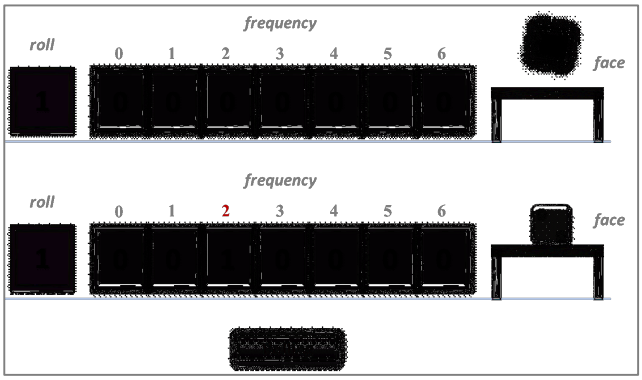


*Figure 6.* Trial 1 of the execution of the dice rolling problem-solving learning object.

For more illustration, the second dice rolling experiment is shown in Figure 7, which is performed by pressing the *Roll* control button for the second time. The *roll* box is holding the current loop iteration and the dice face is used as an entry index to increment the counter inside box number 1.

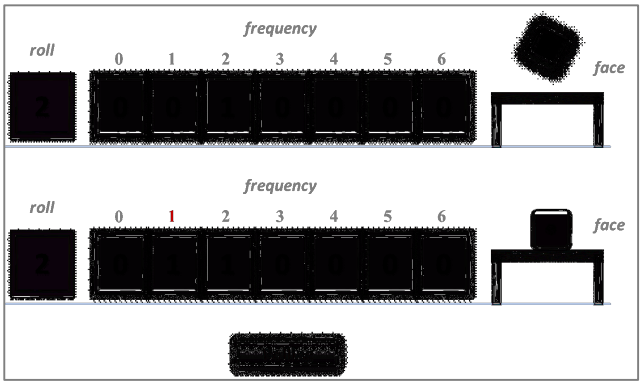


*Figure 7.* Trial 2 of the execution of the dice rolling problem-solving learning object.

Rolling the dice 10 times will generate 10 random outcomes. Based on the performed experiments, the outcomes of 10 rolls are listed in Table 4.

Table 4

*Outcome of Dice Rolling Learning Object for 10 times*

| Face | 1 | 2 | 3 | 4 | 5 | 6 | Total |
|---|---|---|---|---|---|---|---|
| Frequency | 1 | 2 | 1 | 1 | 2 | 3 | 10 |

Figure 8 illustrates the last frame after conducting all the iterations generated from the learning object. The sum of the counters' values reflects the total number of experiments that have been performed, and, consequently, the number of iterations that have been completed by the loop code.

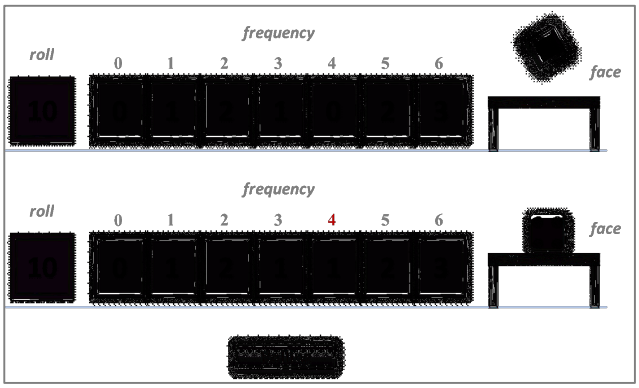


*Figure 8.* State at the end of iteration 10 of execution of the dice rolling problem-solving learning object.

After the completion of the tenth experiment, the *roll* box, which holds the loop control variable, is incremented by one according to the loop header code. That value corresponds to the loop iteration value, which makes the loop condition to be false. The loop control variable is 11, which indicates the completion of the experiments, so, the falseness of the stopping-condition, which will lead to loop termination. This situation is shown in Figure 9. The control button will show the learning object termination by disabling the button and displaying the message *Done*.

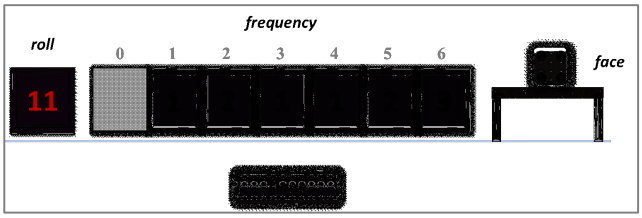


*Figure 9.* Learning object setting after termination.

Using the problem-solving learning object several times is possible. This indicates the possibility of learning object reusability, and thus it enhances the learning process by emphasizing the same information as much as the learner needs. The learner can benefit from using the implemented learning object. Usually the leaner only interacts with the code as given in Figure 3. However, using the learning object in addition to displaying the programming code illustrates the concepts with the support of animations and graphics, as shown in Figure 4. It is expected that this learning object exhibit will better stimulate the learner's sensory learning.

## Conclusion

This paper demonstrates an approach for designing learning objects for programming-based problem solving using animations and computer graphics. Practices and experiences from the teaching and delivery of related courses were adopted in the design approach.

The main features of this approach are its efficiency and support in teaching and learning the programming concepts used in problem-solving techniques. The implemented outcomes can function as practical experience for the learners of the subject matter. The developed learning objects are characterized by reusability and the possibility of integration into e-learning environments. The technique applied for implementing the learning objects for computer programming-based problems can be extended to other fields and disciplines of science and technology.

Appropriate presentation of the learning objects in the teaching-learning process is important to encourage learners' frequent usage. However, the usage expected will be based on the learners' learning styles. Learning objects are not the only resources we might rely on in teaching programming, but could be useful if integrated into teaching and learning as a natural part of the learner's programming experience.

As future work, we intend to use this approach to implement more learning objects for problem solving of a different nature to refine the model so that it can better fit other related learning paradigms, such as flipped learning, active learning, and m-learning. Further analysis of the effects of the use of the learning objects in teaching computer-based problem solving can be conducted and investigated. It is intended to use the proposed application with an experimental group and report any improvement in the learning process compared to a control group that will only use the traditional code segment.

References

Adamchik, V., & Gunawardena, A. (2003). A learning objects approach to teaching programming. *Proceedings ITCC 2003: International Conference on Information Technology: Coding and Computing* (pp. 96-99). Las Vegas, NV: Institute of Electrical and Electronics Engineers (IEEE). doi:10.1109/ITCC.2003.1197507

Anderson, I. (2016). Identifying different learning styles to enhance the learning experience. *Nursing Standard, 31*, 53-63. doi:10.7748/ns.2016.e10407

Andreza, B., & Magalhães, F. (2019). Inclusive model application using accessible learning objects to support the teaching of mathematics. *Informatics in Education, 18*(1), 213-226. doi:10.15388/infedu.2019.10

Awla, H. (2014). Learning styles and their relation to teaching styles. *International Journal of Language and Linguistics, 2*(3), 241-245. doi:10.11648/j.ijll.20140203.23

Begosso, L., Begosso, L., & Begosso, R. (2016). An approach for the use of learning objects in teaching computer programming concepts. *2016 IEEE Frontiers in Education Conference (FIE)* (pp. 1–8). Erie, PA: IEEE. doi:10.1109/FIE.2016.7757619

Begosso, L., Begosso, L., Begosso, R., Ribeiro, A., & Martins, R. (2015). The use of learning objects for teaching computer programming. *2016 IEEE Frontiers in Education Conference (FIE)* (pp. 786–791). Erie, PA: IEEE. doi:10.1109/FIE.2015.7344148

Cavus, N., & Ibrahim, D. (2004). Using learning objects to teach programming languages. *Creating the Future 3rd FAE International Symposium* (pp. 303-308). Lefke, Cyprus: European University of Lefke. Retrieved from https://files.eric.ed.gov/fulltext/ED503158.pdf

Chiappe, A., Cifuentes, Y., & Rodríguez, H. (2007). Toward an instructional design model based on learning objects. *Educational Technology Research and Development, 55*, 671-681. doi:10.1007/s11423-007-9059-0

Falvo, D. (2008). Animations and simulations for teaching and learning molecular chemistry. *International Journal of Technology in Teaching and Learning, 4*(1), 68–77. Retrieved from https://sicet.org/main/wp-content/uploads/2016/11/ijttl-08-01-4_1_5_Falvo.pdf

Ford, J., Robinson, J., & Wise, M. (2016). Adaptation of the Grasha Riechman Student Learning Style Survey and Teaching Style Inventory to assess individual teaching and learning styles in a quality improvement collaborative. *BMC Medical Education, 16*. doi:10.1186/s12909-016-0772-4

Halverson, R., Wolfenstein, M., Williams, C., & Rockman, C. (2009). Remembering math: The design of digital learning objects to spark professional learning. *E-Learning and Digital Media, 6*(1), 97–118. doi:10.2304/elea.2009.6.1.97

Henderson, W. (1994). Animated models for teaching aspects of computer systems organization. *IEEE Transactions on Education, 37*(3). doi:10.1109/13.312133

Hoffler, T., & Leutner, D. (2007). Instructional animation versus static pictures: A meta-analysis. *Journal of Learning and Instruction, 17*, 722–738. doi:10.1016/j.learninstruc.2007.09.013

Hughes, G. (2016). Identifying learning styles. *Nursing Standard, 31*(16-18), 72-73. doi:10.7748/ns.31.16-18.72.s49

Hwang, I., Tam, M., Lam, P., & Lam, S. (2012). Review of use of animation as a supplementary learning material of physiology content in four academic years. *The Electronic Journal of e-Learning, 10*(4), 368–377. Retrieved from http://www.ejel.org/issue/download.html?idArticle=216

IEEE (2002). IEEE Standard for Learning Object Metadata (*IEEE Std 1484.12.1-2002)*. doi:10.1109/IEEESTD.2002.94128

Jaimez-González, C., García-Mendoza, B., Luna-Ramírez, W., Nápoles-Duarte, M., & Vargas-Vargas, A. (2018). Learning objects to support the teaching-learning process of a web fundamentals undergraduate course. *American Journal of Educational Research, 6*, 1573-1580. doi:10.12691/education-6-11-17

Krassimir, K., & Lu, I. (2017). On the possibility of preferred performance styles and their link to learning styles. *Frontiers in Education, 2*. doi:10.3389/feduc.2017.00032

Lillejord, S., Borte, K., Nesje, K., & Ruud, E. (2018). *Learning and teaching with technology in higher education - a systematic review.* Oslo: Knowledge Centre for Education, The Research Council of Norway. Retrieved from https://www.forskningsradet.no/siteassets/publikasjoner/1254035532334.pdf

Lowe, R.K. (2004). Animation and learning: Value for money? In R. Atkinson, C. McBeath, D. Jonas-Dwyer & R. Phillips (Eds), *Beyond the comfort zone: Proceedings of the 21st ASCILITE Conference* (pp. 558-561). Perth, Australia: Australian Society for Computers in Learning in Tertiary Education. Retrieved from http://www.ascilite.org.au/conferences/perth04/procs/lowe-r.html

Luna-Ramírez, W. A., & Jaimez-González, C. R. (2014). Supporting structured programming courses through a set of learning objects. *International Conference on Information Society (i-Society 2014)*. London, UK: Infonomics Society. doi:10.1109/i-Society.2014.7009024

Maseleno, A., Hardaker, G., Sabani, N., & Suhaili, N. (n.d.). Data on multicultural education and diagnostic information profiling: culture, learning styles and creativity. *Data in Brief, 9*, 1048–1051. doi:10.1016/j.dib.2016.11.024

Matthiasdottir, A. (2006). Usefulness of learning objects in computer science learning. The Codewitz project. *Proceedings of the Codewitz Open Conference Methods, Materials and Tools for Programming Education* (27-31). Tampere, Finland. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download? doi:10.1.1.559.1156&rep=rep1&type=pdf

Narasimhamurthy, U., & Shawkani, K. A. (2009). Teaching of programming languages: An introduction to dynamic learning objects. *2009 International Workshop on Technology for Education* (114-115). Bangalore, India: IEEE. doi:10.1109/t4e.2009.5314123

Osman, W., & Elmusharaf, M. (2014). Effectiveness of combining algorithm and program Animation: A case study with data structure course. *Issues in Informing Science and Information Technology, 11*, 155-168. doi:10.28945/1986

Pashler, H., McDaniel, M., & Rohrer, D. B. (2008). Learning styles: Concepts and evidence. *Psychological Science in the Public Interest, 9*(3), 105-119. doi:10.1111/j.1539-6053.2009.01038.x

Rotbain, Y., Marbach, G., & Stavy, R. (2008). Using a computer animation to teach high school molecular biology. *Journal of Science Education and Technology, 17*(1), 49-58. doi:10.1007/s10956-007-9080-4

Smith, G., & Escott, E. (2004). Using animations to support teaching of general computing concepts. *Proceedings of the Sixth Australian Computing Education Conference (ACE2004)* (304-301). Dunedin, New Zealand: Australian Computer Society, Inc. Retrieved from https://crpit.scem.westernsydney.edu.au/confpapers/CRPITV30Smith.pdf

Tapoli, P., & Mikropoulos, T. (2019). Digital learning objects for teaching computer programming in primary students. In M, Tsitouridou, J. A. Diniz, & T. A. Mikropoulos (Eds.), *Technology and innovation in learning, teaching and education* (pp. 256-266). TECH-EDU 2018. Communications in Computer and Information Science, vol 993. Switzerland: Springer. doi:10.1007/978-3-030-20954-4_19

Villalobos, J. C., & Jiménez, C. (2009). Developing programming skills by using interactive learning objects. *ITiCSE '09 Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science* (pp. 151-155). New York, USA: ACM. doi:10.1145/1595496.1562927

Wiley, D. A. (2000). Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. In D. A. Wiley (Ed.), *The instructional use of learning objects: Online version*. Retrieved from: http://reusability.org/read/chapters/wiley.doc

Willingham, D., Hughes, E., & Dobolyi, D. (2015). The scientific status of learning styles theories. *Teaching of Psychology, 42*(3), 266-271. doi:10.1177/0098628315589505

**Author**

Abdullah Basuhail, Professor of Computer Engineering, Computer Science Department, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia. Email: abasuhail@kau.edu.sa