Game–Theme Based Instructional Module for Teaching Object Oriented Programming

Sharad Sharma, Senior Member, IEEE, James Stigall, Sarika Rajeev
Department of Computer Science
Bowie State University
Bowie, MD 20715, USA
Email: ssharma@bowiestate.edu, {stigallj0813, rajeevs1027}@students.bowiestate.edu

***Abstract*— Due to complex mathematical nature of computer science, it is considered a complex and arduous subject by college students. Due to a gradual decrease in computer science students, in spite of a growing demand for computer science professionals, it is crucial to find a way to attract computer science students by making the concepts even more fascinating and absorbing. The aim of this paper is to develop game theme based instructional modules for computer science students that motivates and engages students while contributing to their learning outcomes. Game theme based instructional modules are designed to encourage faculty to teach and motivate students to learn the concepts of object oriented programming using interactive, graphical, game-like examples. This paper discusses the design parameters and implementation of the module and describes a case study of adopting the module in an existing class. The results of the study demonstrate the effectiveness of the instructional module and the possibility to include it in the existing curriculum with minimum alterations to the existing established course material. The instructional modules act as a supplement to an existing course and enable faculty to explore teaching with a game-theme materials and helping students to be more motivated and engaged in class.**

Keywords—Games;Courseware; Assessment;

I. INTRODUCTION

Game Theme based Instructional (GTI) modules are widely used in academia as supplementary teaching tools with the expectation that students will be able to master the content within the modules. These types of software applications are typically built around the notion that the student can learn a concept or idea by experiencing it in real life. Educational gaming modules are also built with usability in mind whereas users should find the modules to be intuitive.

Ever since the first game [1] came into existence, we have observed dramatic increase in the popularity of computer games. In recent years, there has been a considerable interest in the gaming modules, when a game was added as a flavor for computer science students [2]. There are already many studies pertaining to the use of games for education, and most of them agree that software games provide tangible, understandable, and appealing context in learning of computer science [3, 4, 5]. Active learning is a kind of learning procedure, when students use some interactive instrument such as [2,3] gaming modules to learn the concepts. It is clear that when anyone is actively engaged in learning, then it is beneficial for learning. A commitment to the learning environment facilitates exploration is the subject while engagement promotes the need to explore further [6]. Existing research and studies [7, 8] have shown that the game development theory has been successful for CS1 courses, but there are relatively very few studies which concentrate on CS2. CS2 is still a gridlock for computer science learning.

In this paper we show game theme based instructional modules developed for computer science students. The module demonstrates the concepts of object oriented programming language (OOP) specifically, encapsulation, polymorphism, and inheritance. The module was implemented using Python within Vizard, a virtual reality development toolkit. Three-dimensional (3D) models featured in the module were created in 3ds Max, a 3D modeling and animation software application. The results demonstrate that the module helped students learn OOP concepts and found the module user-friendly. During the semester, the module was evaluated with undergraduate students in a programming course. The students experimented with the module and took a survey afterwards. Survey data demonstrates that the module makes a significant educational impact on students.

Fig. 1. Virtual Instructor used in the OOP instructional module.

Game-themed instructional modules have the potential to help students understand these concepts as they present life-like challenges and prompt students (or users) to come up with solutions associated with those challenges and the concepts they are trying to learn. The rest of the paper is structured as

978-1-4673-9795-7/15 $31.00 © 2015 IEEE
DOI 10.1109/CSCI.2015.35

follows: Section II, describes the previous work briefly. Section III, illustrates the design of OOPS module. In Section IV, we discuss its implementation using the Vizard Virtual Reality Toolkit. In Section V, we evaluate the effectiveness of the two modules with a class of undergraduate computer Science majors.

II. Related Work

A. Educational Software

Educational software helps students learn concepts by applying classroom theory to real-world events. This can be seen as *constructivsm* where a student acquires, or "builds", knowledge from real-world applications of things learned in the classroom [9]. According to Tam [10], constructivism and technology provides the student with opportunities to identify problems and to use critical thinking to solve those problems. Vichido, et al. [11] state that critics of constructivism regard it as nothing more than a trend and it is unclear that courses using the approach should yield the same degree of achievement as those using more traditional approaches. An instructional software application teaching Japanese was evaluated by Nagata [12] with fourteen students, whose first language was not Japanese. An educational software application evaluated by Homer, et al. [13] increased children's knowledge of asthma, the main character of which suffers from it. The group that used the software got 77% of the post-test questions correct compared to 63% correct from those studying only the paper-based material.

B. Usability in Educational Games

Usability is an important aspect of instructional software. Costible, et al. [14] states that if instructional software is not usable, then users will spend more time trying to understand how to use it instead of learning the content within the software – poor usability distracts students from learning objectives. Koutanis, et al. [15] and Virvou, et al. [16] defined usability issues faced in developing educational software. Koutanis, et al. argued that the software aspects that users pay the most attention to are the objectives of the application, installation of additional software on their computers, protection of personal information by the software, content completeness within the software, reaction time of the user selection, and references to information sources in the software. Chang, et al. [17] found that users who regularly play video games tend to be more engaged and less frustrated than those who do not regularly play video games. Ahmad, et al. [18] also evaluated a mobile application that teaches English as a foreign language to school-age children by surveying parents and teachers and interviewing the children using it.

C. Educational Virtual Reality (VR) Games

Educational virtual reality (VR) games motivate students to learn. According to Psotka [19], combining VR with educational games helps students understand complex events and objects using abstractions and real-life experiences. They also foster teamwork and social skills when played in a multi-user environment as observed by Martinez-Reyes and Hernandez-Santana [20]. In an age where a majority of youth are familiar with VR technologies such as video games, its role in education is relevant as stated by Callaghan, et al. [21] and by Burkle and Kinshuk [22]. Educational VR games should be considered as an alternative to a more traditional curriculum when the real environment being studied is harmful to the student or to itself, if the student interacts with it [23]. Cecil, et al. [24] evaluated a virtual reality module developed for a course in micro assembly. On the other hand, Yahaya [25] assessed a module that taught decision making to students majoring in Business and found that the students using the module were engaged in the tasks presented to them and fully understood the decision-making processes that helped them complete those tasks. Also, Mavrogeorgi, et al. [26] developed a mobile VR application that provided the user with information on historical sites when the user was in front of it.

VR applications and educational modules are also seen in industry. Akiyoshi, et al. [27] created a module to train employees on power equipment maintenance in an effort to help visualize the equipment and to deepen their understanding of maintenance tasks. A VR system teaching metal casting was implemented by Watanuki and Kojima [28].

III. Modeling and Design of Game theme instructional module (GTI)

This section discusses the modeling of the OOP instructional module, gives a brief overview of the Vizard platform used to develop the games, and the concepts covered in the module.

A. Design Considerations

The modules were built with the Constructivist Theory in mind, which states that the student "builds" knowledge by experiencing it for themselves in the real world. The modules were also built using functional and nonfunctional requirements. The functional requirements were taken from the student's perspective while the nonfunctional requirements were taken from the instructor's perspective. The functional requirements are listed as follows:

- *User interface must be intuitive* – students using any of the modules should not have any difficulty interacting with them. This requirement makes user experience enjoyable.
- *The student should be able to restart either module at any time* – if a student experiences a technical difficulty (e.g.: a module crashes, not getting correct responses), then the module should be restarted.
- *The student should be able to understand the instructions* – user instructions detailed in each module should be easy to understand so that the student can navigate through the module without confusion.
- *Pseudocode displayed should be comprehensible* – as with the instructions, pseudocode should also be easy to understand. The purpose of it is to give the student a high-level view of how a polymorphism, inheritance, and encapsulation operates.

- *Within each module, the student should be able to switch between one game and another* – if a student grows tired of a game within any module, the student should switch from that game to another without difficulty.

The nonfunctional requirements are listed as follows:

- *Each module should motivate the student to learn* – the objects (pseudocode, graphics, etc.) featured in each module should spawn interest in the student for programming. Each module should motivate the student to further his/her knowledge of loops, arrays, and other data structures.
- *Each module should teach the student about the subject* – although each module features examples from disciplines outside of Computer Science, the module should ultimately teach the student about object oriented programing concepts. The examples featured in each module should not deviate from the subject matter.
- *Reaction to user input should be immediately rendered* – the student should not wait for any reaction to his/her input (i.e. button clicks). Waiting for reaction could impair user experience.
- *Graphics should be appealing to the student* – each module should feature graphics (3D models, buttons, text, etc.) that visually appeal to the user. This requirement enhances user experience.
- *The modules should be portable* – the student should be able to play each module on any platform. That is, it should work whether the student is using a Windows or Macintosh system.
- *An award system should be featured* – having an award system would give the student some idea of his/her learning progress for OOP.

The environment for the module was built using three-dimensional models found on various websites. The models were adapted into a format compliant with Vizard using 3ds MAX, a software application where 3D models can be created and animated. Afterwards, textures were added to the lake, grass, and gazebo area. The environment for the module includes a matrix of buttons and a built-in male instructor avatar as shown in fig. 1.

B. Vizard Framework

Vizard Virtual Reality Toolkit is a Python-based integrated development environment (IDE) used to develop virtual reality applications. Three-dimensional (3D) models can be built in 3ds MAX and then imported into the Vizard environment using its built-in exporter. Picture (.jpg or .png) files can be also be incorporated into the environment. Models and pictures imported into Vizard can be positioned into the environment and scaled to fit it. Through its libraries, Vizard provides built-in functions that govern interactions between objects and their environments. One can also add shapes, text, buttons, and sliders through those functions. Vizard consists of a Python script editor and debugger. After a script has been created, it can be run with or without debugging. It can also be published as an executable (.exe) file for use by the general public.

IV. Game theme instructional module (GTI) Implementation

The development of the Object-Oriented Programming (OOP) training module aimed to supplement the curriculum surrounding OOP, whether it is used as an in-class activity or as a study tool. The design and implementation phase consisted of creating the module with the user in mind.

A. OOP Instructional Module

The OOP training module was developed with two goals in mind: instruction and user-friendliness. Each game in the module starts with a tutorial on the game's topic (for example, the Polymorphism game would start with a tutorial on polymorphism). The tutorials start by explaining what the topic is and then gives examples to further help the user understand the topic. Tutorials were built with beginners in mind but users possessing more advanced knowledge on the topic may skip a tutorial and proceed directly to the game. The first goal was also met through the Constructivism Theory – where one "builds" knowledge by practicing what is being studied. The quizzes and challenges featured in each game in the module embodies this theory by enabling the student to construct knowledge through gaming metaphors.

As far as usability is concerned, user instructions given throughout the module are clear so that the user is able to understand the objectives of the game and the concept being taught. Also, at the end of each game, the user can either play the game again or play another game. If the user plays the game again, the user has the opportunity to improve his/her performance during the second attempt, provided that the user did not play the game as well as expected the first time. The module's graphics (3D models, text, pictures, etc.) were added to make the module visually appealing to the user, enhancing the user's experience and actualizing the game-like environment. Lastly, the module is portable – meaning that it can be played on any platform (e.g. Windows, Macintosh) providing convenience for the user.

B. MVC Implementation

The OOP training module was developed using the Vizard Virtual Realty Toolkit – a Python-based integrated development environment (IDE) used to create VR applications. Three-dimensional models featured in the module were developed using 3ds Max 2014 and imported into Vizard, and scaled to fit the user environment. Picture files were also imported and scaled. The functionalities for those models, pictures, and user interface elements (buttons, text boxes, etc.) were implemented in Python. Built-in avatars were used to interact and engage the user.

The software architecture pattern used is illustrated in Fig. 2. The model-view-controller (MVC) architecture was used to develop the model whereas the *model* represents the data, the *view* represents how the data is presented to the user, and the *controller* represents how the data is manipulated by the user.

In this module, the Python source code is the *model* in the software architecture. The renderer used to generate the scenes, the 3D models seen in the model, and the shapes and figures also seen in the model all constitute the *view* of the module's architecture. The mouse and keyboard commands and buttons are used to interact with the module, so they constitute the *controller* in the architecture.

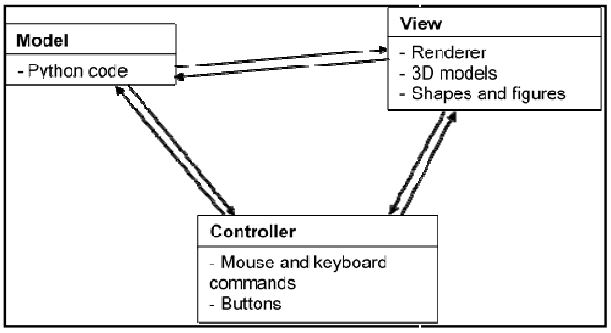

Fig. 2. Software architecture used in developing the module

All the elements in the *view* (except for the renderer) are defined by the source code, which also defines the available commands and the buttons. Since the *controller* elements are used to interface with the module, they control what part of the source code can be executed and which 3D model and other screen elements are visible at a given time. The visibility of the *view* elements control what commands and what portion of the code can be executed.

C. GTI Modules

The module is divided into three games, each discussing one topic related to object-oriented programming, namely: inheritance, polymorphism, and encapsulation.

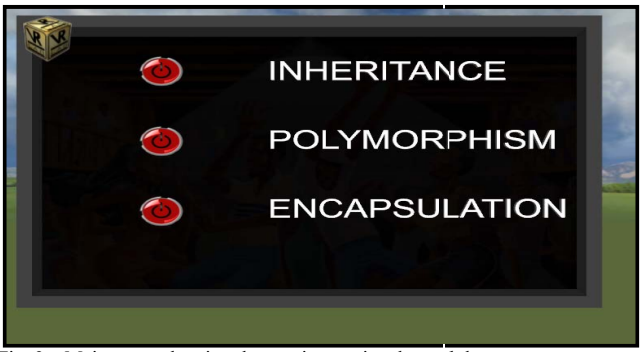

Fig. 3. Main menu showing the tree instructional moodules

When the user starts the module, he/she is greeted with an avatar giving them an overview of what it is about followed by the main menu. The welcome screen and main menu can be seen in Fig. 2 and 3.

1) Inheritance Module

In this module, the user is greeted with a UML (unified modeling language) class diagram located in the top right of screen. That diagram has only one class, Vehicle, and one object, the integer variable "speed". At the start of the game, the user is asked to build a vehicle by first choosing what type of vehicle he/she wants to build – an air vehicle or a land vehicle. The UML diagram expands to include the class corresponding to the type of vehicle the user wishes to build. The newly added class is shown to be the child class of the Vehicle class and, thus, inherits the properties of the Vehicle class. Lastly, the user is asked to choose the purpose of the vehicle – whether it will be used for military purposes or for commercial purposes. Once the user makes that choice, the UML diagram expands again to include the class corresponding to the chosen purpose for the vehicle. That class is displayed as being the child class for the previously-chosen vehicle type class – the purpose class inherits the properties of the type class. The resulting vehicle is displayed, as well.

A screenshot of the Inheritance module is shown in Fig. 4. In the screenshot, the parent class is vehicle and the child class is Air, corresponding to the type of vehicle. The class Air inherits the variable, "speed", from Vehicle while containing its own function, fly(). The child class for Air is the class Commercial, which inherits two objects from Air, "speed" and fly(), while containing its own objects: the variable "Airline" and ArrivalT(). The vehicle that ends up being built is a commercial jetliner.

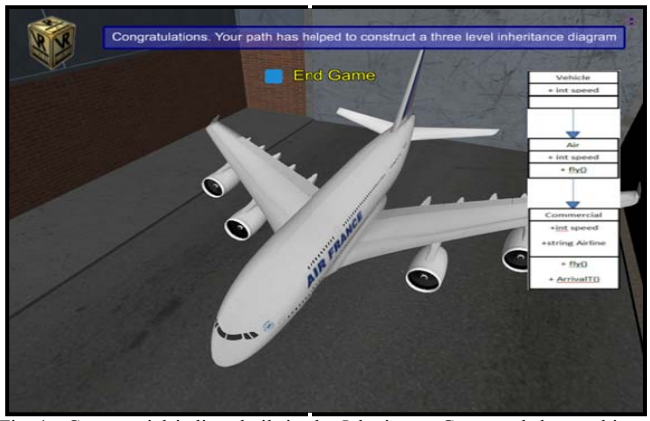

Fig. 4. Commercial jetliner built in the Inheritance Game and the resulting UML diagram.

2) Polymorphism Module

In the Polymorphism module, the user is asked to guess a number between 1 and 20, then the computer guesses a number, also between 1 and 20. If either the user or the computer guesses correctly, the game is over. If the guess is incorrect, either player loses two points. Both, the user and the computer start out with 10 points, so the game is over when both players make five incorrect guesses (for a total of zero points) or if either player makes a correct guess.

In Fig. 5, the user is represented by an avatar, named Jack, located to the left of the screen. The computer is represented by a picture of a laptop, located at the right of the screen. Beneath each player is Java pseudocode describing how each player is making their guesses. Jack (the user) makes the guess by using his brain while the computer uses a random number generator to make the guess. Both pieces of pseudocode, the one located below Jack and the one located below the computer, contain a common function, getGuess(). However, that function behaves differently. This metaphor

realizes the principle of polymorphism where multiple classes can have the same function(s) but they may behave differently.

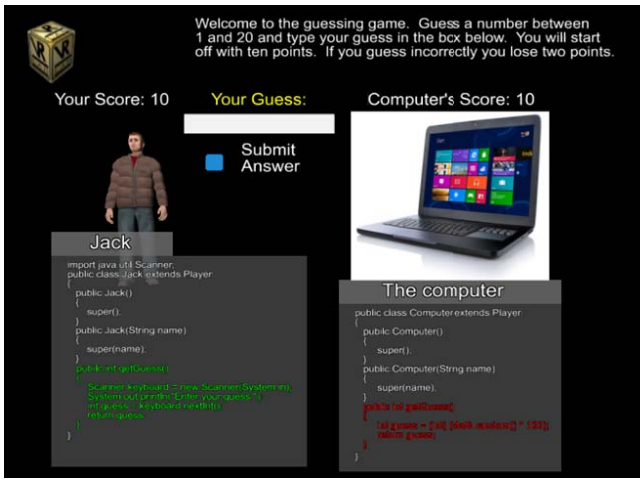

Fig. 5. Screenshot of the Polymorphism Game

3) Encapsulation Module

The Encapsulation module is comprised of two quizzes. In the first quiz, the user is asked to assign the proper access modifier (those being public, private, or protected) to a certain variable or function in the class Car.

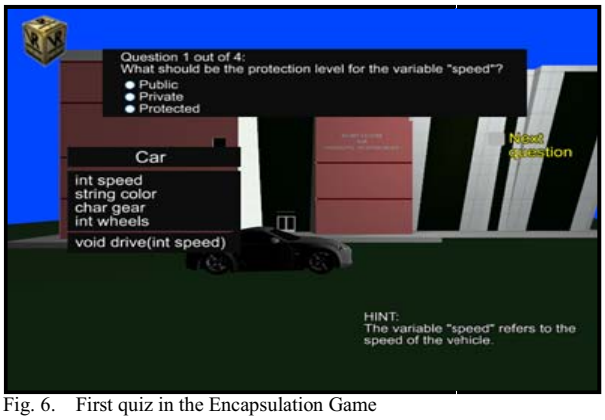

Fig. 6. First quiz in the Encapsulation Game

For each object in the class Car, the user must respond in terms of the real-life operation of a car. Once the quiz is done, the user is given feedback on his/her answers and is awarded points. The user may take a second quiz or quit the game. A screenshot of the first quiz can be seen in Fig. 6.

V. Results and Analysis

Game theme based instructional modules have been analyzed with interaction of students to these modules. During the fall semester, the OOP module was evaluated with twenty undergraduate students in a CS2 programming course. The students completed the pre and post survey. All the students had taken a previous CS1 programming course. 80% students were male and 20% students were female. All the students were computer science and technology majors, while one student was a mathematics major. The students experimented with the module and took a survey afterward. Survey data demonstrates that the module makes a significant educational impact on student

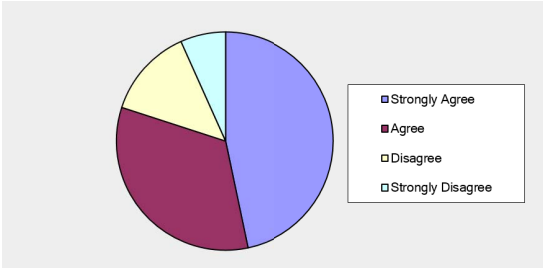

Fig. 7. The OOPS game developed my interest in programming/gaming

As seen from figure 7, majority (80%) of students agreed that the use of OOPS instructional module developed their interest in programming and gaming. On the other hand as seen in figure 8, 86.99% students felt that they were able to understand the code (Inheritance, Polymorphism, and Encapsulation) while playing the OOPS game.

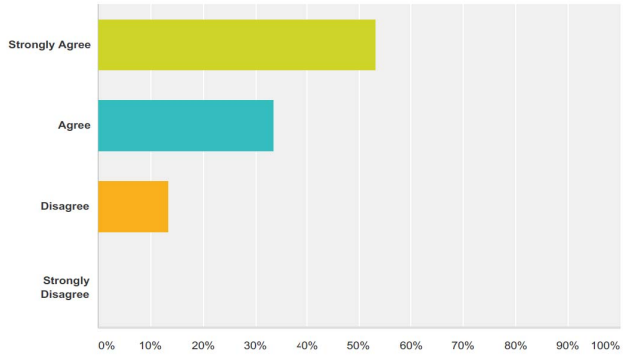

Fig. 8. Were you able to understand the code(Inheritance, Polymorphism, and Encapsulation) while playing the OOPS game?

Results from the survey given after the instructional module was used in class reveal that students were generally satisfied with each module in the areas of clarity (92%) in write-up and the graphical user interface (90%).

VI. Conclusion

We have designed and implemented GTI modules for teaching OOPS concepts aiming to provide an alternative way of teaching the students in introductory Computer Science courses. We evaluated the GTI module in a class of twenty undergraduate Computer Science major students. Survey data demonstrates that the module makes a significant educational impact on student learning concepts.

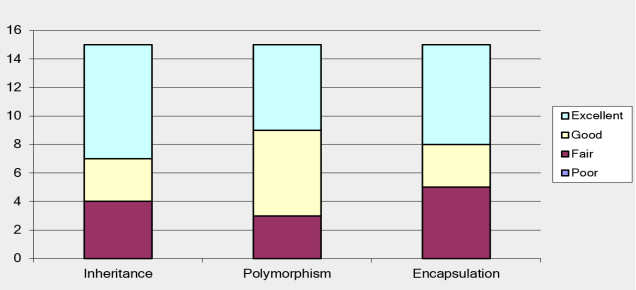

Fig. 9. How would you rate the following learning concepts?

The modules allowed the instructor to use GTI and to improve his own understanding of game-like programming with a minimal time investment. This allowed the instructor to increase his expertise and confidence in building his own game-them based teaching material.

ACKNOWLEDGMENT

The authors would like to thank the National Science Foundation for supporting the project. This work is funded under the grant award number- HRD-1238784.

REFERENCES

[1] Pooh. W . Amy, "Computer game addiction and emotional dependence", 2012.

[2] Bayliss, Jessica D and Strout, Sean. Games as a "Flavor" of CS1. Proceedings of the 37th SIGCSE technical symposium on Compute r science education, ACM, pp. 500-504,Houston,Texas, USA, 2006.

[3] Leutenegger, Scott and Edgington, Jeffrey.ACM SIGCSE Bulletin Volume 39 , Issue 1 , pp. 115-118.A games first approach to teac hing introductory programming. 2007.

[4] Amory, A., Naicker, K., Vincent, J. & Claudia, A. "Computer games as a learning Resource". Proceeding of ED-MEDIA, ED-TELECOM, World conference on Education Multimedia and educational Telecommunications, vol.1, pp.50-55, 1998.

[5] Conati, C. & Zhou, X. Modeling student's emotion from cognitive appraisal in educational games.", 2002.

[6] Price, S.Rogers, Y.Scaife, M.Stanton, D. & Neale, H. "Using tangibles to promote novel forms of playful learning" In interacting with computers (2003), pp. 169-185, 2003

[7] Garlick, Ryan and Akl, Robert. 9th International Conference on En gineering Education.Interclass Competitive Assignments in CS2: A One Year Study, 2006.

[8] Forte, Andrea and Guzdial, Mark. Big Island, Hawai`i: IEEE Com puter Society.Proceedings of the Proceedings of the 37th Annual H awaii International Conference on System Sciences (HICSS'04) - T rack 4 - Volume 4. p. 40096.1.Computers for Communication, Not Calculation:Media as a Motivation and Context for Learning, 2004.

[9] S. Christina and K. Dimitrios, "Applying Constructivism for Interactive Educational Software: a Research Based Design, Implementation and Evaluation Method" in Proc. IEEE International Conference on Advanced Learning Technologies (ICALT '04), 2004 © IEEE. doi: 10.1109/ICALT.2004.1357614.

[10] M. Tam, "Constructivism, Instructional Design, and Technology: Implications for Transforming Distance Learning", Educational Tech. & Society, vol. 3, no. 2, Apr. 2000.

[11] C. Vichido, M. Estrada, et al., "A constructivist educational tool: Software architecture for web-based video games" in Proc. Fourth Mexican Int. Conf. on Computer Science (ENC '03), IEEE. doi: 10.1109/ENC.2003.1232888, 2003.

[12] N. Nagata, "Input vs. Output Practice in Educational Software for Second Language Acquisition", Language Learning and Technology, vol. 1, no. 2, pp. 23-40, Jan. 1998.

[13] C. Homer, O. Susskind, et al., "An Evaluation of an Innovative Multimedia Educational Software Program for Asthma Management: Report of a Randomized, Controlled Trial", Pediatrics: Official Journal of the American Academy of Pediatrics, vol. 106, no. 1, pp. 210-215, Jul. 2000.

[14] M. F. Costabile, M. De Marsico, et al., "On the Usability Evaluation of E-Learning Applications", IEEE. doi: 0-7695-2268-8/05, *Proc. 38th Hawaii Int'l Conf. on System Sciences*, 2005.

[15] D. Koutanis, M. Virvou, et al., "Identifying and Managing Risks in the Development of Online Educational Software" in *5th International Conference on Information, Intelligence, Systems and Applications, IISA 2014*, Chania, Greece, pp. 245-252, 2014.

[16] M. Virvou and G. Katsionis, "On the usability and likeability of virtual reality games for education: The case of VR-ENGAGE", *Computers & Education*, vol. 50, no. 1, pp. 154-178.

[17] M. Chang, M. Evans, et al., "Educational Video Games and Students' Game Engagement" IEEE. doi: 10.1109/ICISA.2014.6847390, in *2014 Fifth Int. Conf. on Information Sci. and Applications (ICISA 2014)*, 2014.

[18] W.F.W. Ahmad, A.R.S. Shaarani, et al., "Mobile Language Translation Game" in *Proc. 2012 International Conference on Computer & Information Science (ICCIS)*, Kuala Lumpur, Malaysia, pp. 1099-1104, 2012.

[19] J. Psotka, "Educational Games and Virtual Reality as Disruptive Technologies", *Educational Technology & Society*, vol. 16, no. 2, pp. 69-80, Apr., 2013.

[20] F. Martinez-Reyes and I. Hernandez-Santana, "The Virtual Maze: A game to promote social interaction between children" in *Proc. Eighth International Conference on Intelligent Environments*, Guanajuato, Mexico, pp. 331-334, 2012.

[21] M. J. Callaghan, K. McCusker, et al., "Integrating Virtual Worlds & Virtual Learning Environments for Online Education" in *First International Consumer Electronic Society's IEEE Games Innovation Conference*, London, United Kingdom, pp. 54-63, 2009.

[22] M. Burkle and Kinshuk, "Learning in virtual worlds: The challenges and opportunities" in *Proc. 2009 International Conference on CyberWorlds*, Bradford, United Kingdom, pp. 320-327, 2009.

[23] S. Fan, Y. Zhang, et al., "The Application of Virtual Reality in Environmental Education: Model Design and Course Construction" in *Proc. 2010 Int. Conference Biomedical Eng. and Computer Sci.*, 2010.

[24] J. Cecil, P Ramanathan, et al., "Virtual Learning Environments in Engineering and STEM Education" in *2013 Frontiers in Education Conference*, Oklahoma City, OK, pp. 502-507, 2013.

[25] R. A. Yahaya, "Assessing the Effectiveness of Virtual Reality Technology as part of an Authentic Learning Environment" in *Proceedings of the Sixth International Conference on Advanced Learning Technologies (ICALT '06)*, Kerkrade, The Netherlands, pp. 262-264, 2006.

[26] N. Mavrogeorgi, S. Koutsoutos, et al., "Vivid educational experience with virtual reality" in *Proc. The Fourth International Multi-Conference on Computing in the Global Information Technology (ICCGI '09)*, Cannes/La Bocca, France, pp.196-201, 2009

[27] M. Akiyoshi, S. Miwa, et al., "A Learning Environment for Maintenance of Power Equipment Using Virtual Reality" in *Fifth International Conference on Image Processing and its Applications*, Edinburgh, pp. 331-335, 1995.

[28] K. Watanuki and K. Kojima, "Virtual Reality Based Knowledge Acquisition and Job Training for Advanced Casting Skills" in *Proc. 16th Int. Conference on Artifical Reality and Telexistence,* 2006.