An Investigation into Game Based Learning Using High Level Programming Languages

Ragab Ihnissi
School of Computing and Engineering
University of Huddersfield
Huddersfield, West Yorkshire, UK
e-mail: Ragab.Ihnissi@hud.ac.uk

Joan Lu
School of Computing and Engineering
University of Huddersfield
Huddersfield, West Yorkshire, UK
e-mail: J.lu@hud.ac.uk

***Abstract*— Game-related education within mobile learning spheres is a matter of great debate for university students across the globe. It is the case that programming languages often pose a sizeable challenge for university students. This research paper aims to develop a game based learning platform "iPlayCode", designed to offer a new and exciting method of learning programming language. Xcode 5.0.2 was used to develop the game by using the cocos2d-x development tool and the Adobe Photoshop graphic design tool. In addition, iOS 7.0.3 (11B508) Simulator was used to test the application and the application was deployed in different models of mobile devices such as the iPhone and iPad. The application outcomes are presented by a mobile game that teaches programming languages in an easy, attractive and effective way.**

Keywords-User interface; application; m-learning; mobile game based learning.

I. INTRODUCTION

Mobile technology usage has come of age to such a degree in recent years that now it has surpassed the increase of personal computers in our professional and social lives [1]. Improvements in innovative mobile and wireless technologies have also had a positive impact in our educational settings, thereby creating a new method or means for technology enhanced or improved learning called m-learning (mobile learning) [2]. Mobile technologies provide an opportunity for a fundamental change in education and due to the success of the m-learning community, recently we have noticed a rapid growth in mobile learning in all educational sectors [3]. In addition, game-based learning in m-learning environments has been a subject of interest amongst young people all over the world [4]. Nevertheless, the advancement of digital and mobile technologies has so far been restricted or confined to social communication. However, there is considerable interest in incorporating mobile learning into schools and the potential of these devices for educational use cannot be ignored. The current state of the art games that are used as an auxiliary to learning activities, are restricted to just a single programming language, and they often do not cover the whole vocabulary of this language [5]. The key objective of this project is to develop games that can be played by students who want to build upon and surpass their current level of programming language using their mobile devices. Edutainment is designed to educate, as well as to amuse by adding elements of interest to learning activities and the related contents; game-based learning is a part of edutainment [6]. In recent years, there have been many studies on the use of mobile platforms for education and these include:

A. User Interface (UI)

Bowen and Reeves [7] explored ideas for the improvement of UIs, and aimed to demonstrate ways in which more relaxed perspectives of UIs can be established, based on conventional ideas of improvement.

Several studies focus on usability but ignore the aesthetics of user interfaces. In contrast, many user interface designs emphasise usability yet reduce efficiency and effectiveness. The authors believe that the evaluation of a user interface should depend on the usability, efficiency and effectiveness.

B. Theoretical Studies

Kadyte explored the significance of beginning from the viewpoint of the mobile user, with a theoretical model for enhancing mobile platforms for education [8]. Valk et al [9] assessed support for the function of mobile education, in regards to its impact on the enhancement of learning outcomes in emerging economies.

C. Game-based M-learning

Tan and Liu [10] developed a mobile-based interactive learning environment (MOBILE) to help elementary school students with their English learning. Ab Hamid and Fung outlined a framework, built around the use of mobile gaming devices, designed to aid the comprehension of programming languages [6]. The key objective of this study is to develop games that can be played by students who want to build upon and surpass their current level of programming language using their mobile devices. The iPlayCode project is in its second version and the focus is to ensure that the correct types of games are developed and are suitable to be applied with the domain subject, which includes Objective C, Java for Android, Java, C#, C++ and Python programming languages. In addition, the developer also had to look at the user interface aspect because of the limited screen size of mobile devices. The paper is organised as follows: Section II explains the interface and game design. The implementation is discussed in Section III and the results are presented in Section IV. Section V shows the testing and evaluation of the application and Section VI is the discussion. The conclusion and future work is given in Section VII.

 ISBN: 978-1-61208-416-9

II. USER INTERFACE AND GAME DESIGN

This section represents an effective design of the user interface and the game layout to enhance the iPlayCode application for the users.

A. User Interface

The creators of the game need to maximise the amount of screen space dedicated to learning to compensate for the small screen size of mobile devices.

Ware [11] stated that "effective design should start with a visual task analysis; determine the set of visual queries to be supported by a design, and then the use color, form and space to efficiently serve those queries".

The user interface must, therefore, be easy to navigate if it is to attract and engage users. Results of evaluation and analysis conducted by the researchers indicate that game creators prefer to use bright colours, including purple, blue, cream and light brown (see Figure 3) [12].

Red, black and white are also used to appeal to users and respond to questions. Bright wooden colour is employed for background and to represent the game's levels. The colours help to emphasise the learning content of the application. To maximise usability, the authors of the game unified the background and the user interface to decrease the size and number of screens opened by the application [13]. The single screen features a tab that enables users to select any of the three levels. Once a level has been selected, another screen is displayed that comprises the question and answer part of the application.

This guarantees ease of use. The user is further stimulated by the inclusion of a tally of their score (see Figures 5, 6 and 7).

B. Game Layout

iPlayCode has three main attributes that make 'our case games' inspiring and fun to learn: a requisite level of challenge, use of fantasy and abstractions to make it more interesting, and triggering the curiosity of the player [14].

There are times when fantasy is missing in games, however. Where this is the case, it is preferable to create a multi-level game where factors such as interaction are the main motivation for playing. Interaction is hugely important for games used in lectures as it promotes student participation. iPlayCode also offers sound or music to motivate players to learn programming languages while playing the game. A game session in iPlayCode begins when the player registers his or her name on the start screen. When the game starts, the application displays a main screen that consists of programming language icons; the player then chooses from the programming languages on their mobile phone.

Each programming language has three levels of difficulty: level 1, level 2 and level 3. The level of the challenge in iPlayCode is adjusted by changing the difficulty level of the questions, which are subsequently randomly generated. Every level has a set of sub-functions, each containing ten questions. The questions time-out after a specified number of seconds, and the countdown time is displayed next to the question. To answer the questions, the user clicks either the right answer button or the wrong answer button on the user interface. An additional button helps the player figure out the correct answer. Each question has a ten-second time limit, and every second represents a point. Therefore, the player needs to decide quickly whether the answer is true or false. There are two screens that display the results. The first screen shows the exposure points obtained by the player in the sub-function level. This screen includes three buttons: the first to repeat the same sub-function; the second to return to another level, and the third to move to the second screen, which displays the total points for the three levels. Gold, silver or bronze medals represent the points obtained by the player. There is also a button to return to the main screen so that the user can select another programming language. Figure 1 shows the iPlayCode design structure.

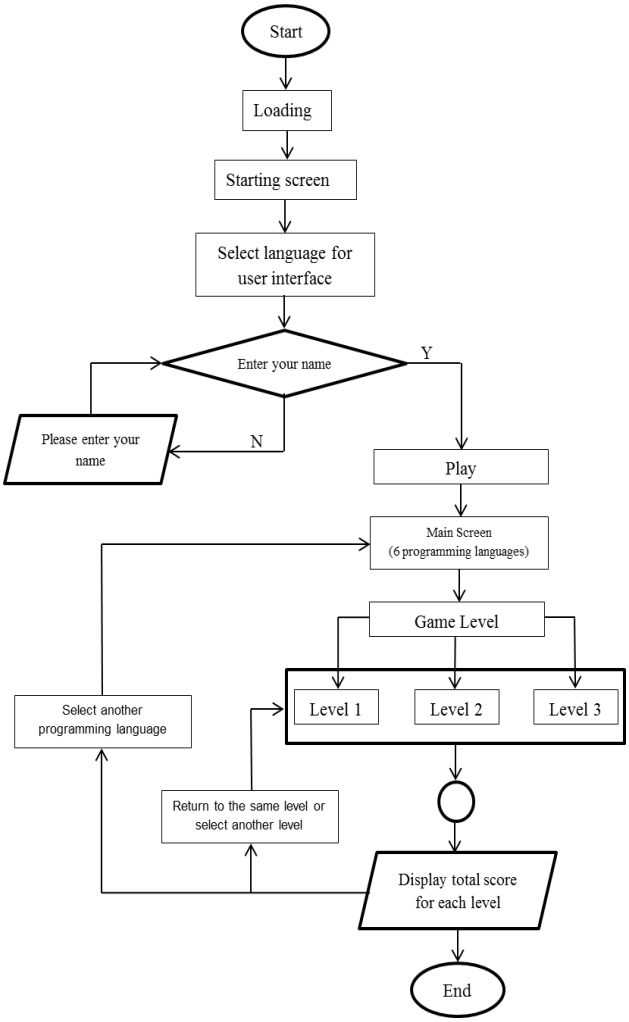


Figure 1. Game flow for iPlayCode

III. IMPLEMENTATION

A. User Interface Implementation

1) *Reduce buttons:* The results and the smaller screen size of mobile phones mean the user interface would benefit from having a limited number of buttons [13], this makes the interface simple to use and creates more space to display essential data. There are three buttons in the user interaction portion of the screen. The CCMenuItemImage class is used to decrease the number of buttons as it includes a menu image and features an upper layer of text in collaboration with CCLabelTTF,

which can be manipulated using the setString method (See Figures 6 and 7). The following code was used to execute this function.

```
Void GamePlay::nextQuestionCallback () {
   RightLabel->setString(Help); //change text
   if (questionNumber < 10) {
      WrongLabel->setString(Next); //change text
   } else {
      WrongLabel->setString(End); //change text }}
```

2) *Minimise pages:* In accordance with the research findings [13], minimal design is used on the application's screens, which includes a main screen that contains three navigation levels. This removes the necessity to create a separate screen for each level (see Figure 5 (a, b and c)).

Navigation between the three levels contains two front and rear backgrounds. The text menu is at the front and controls the contents of the page using CCMenuItemFont class. The background can also be changed to the front to show any of the three levels. The getTag() method obtains the tag of the label for the chosen level. Users can select one of three levels by clicking on the relevant button.

```
void MenuSelect::menuCallback(CCObject* pSender){
CCMenuItemImage* item =
(CCMenuItemImage*)pSender;
if (item->getTag() == 0) { //level1}
else if (item->getTag() == 1) { //level2}
} else if (item->getTag() == 2) { //level3}}
```

B. Game Implementation

To assess the quality of the application and ascertain whether the purpose for which the game was developed has been met, the application was tested for its functionality after development. The game was developed using Xcode 5.0.2 through the cocos2d-x development tool and the Adobe Photoshop graphic design tool. The iOS 7.0.3 (11B508) Simulator was used to test the application. The application was deployed in different models of mobile devices such as the iPhone and iPad.

Although the iPlayCode application implements on mobiles, there is a user interface design challenge for devices that have a small screen. To fully utilise this application these challenges need to be addressed.

Figure 2 shows the interface scrolling up and down through question levels 2 and 3 for each language.

Initially, due to the length of the questions, it was not possible to see the entire question on the iPlayCode display. However, this has been remedied by using the code below, enabling the user to scroll up or down by clicking and dragging the question on the interface.

Figure 2. Scrolling down and up of the questions

```
pScrollView = CCScrollView::create(CCSizeMake(500,
230));
    pScrollView->setContentSize(pQuestionLabel-
>getContentSize());
    pScrollView->setContainer(pQuestionLabel);
    pScrollView->setBounceable(true);
```

IV. RESULTS

The default language of the iPlayCode application is English. However, other languages such as Arabic or

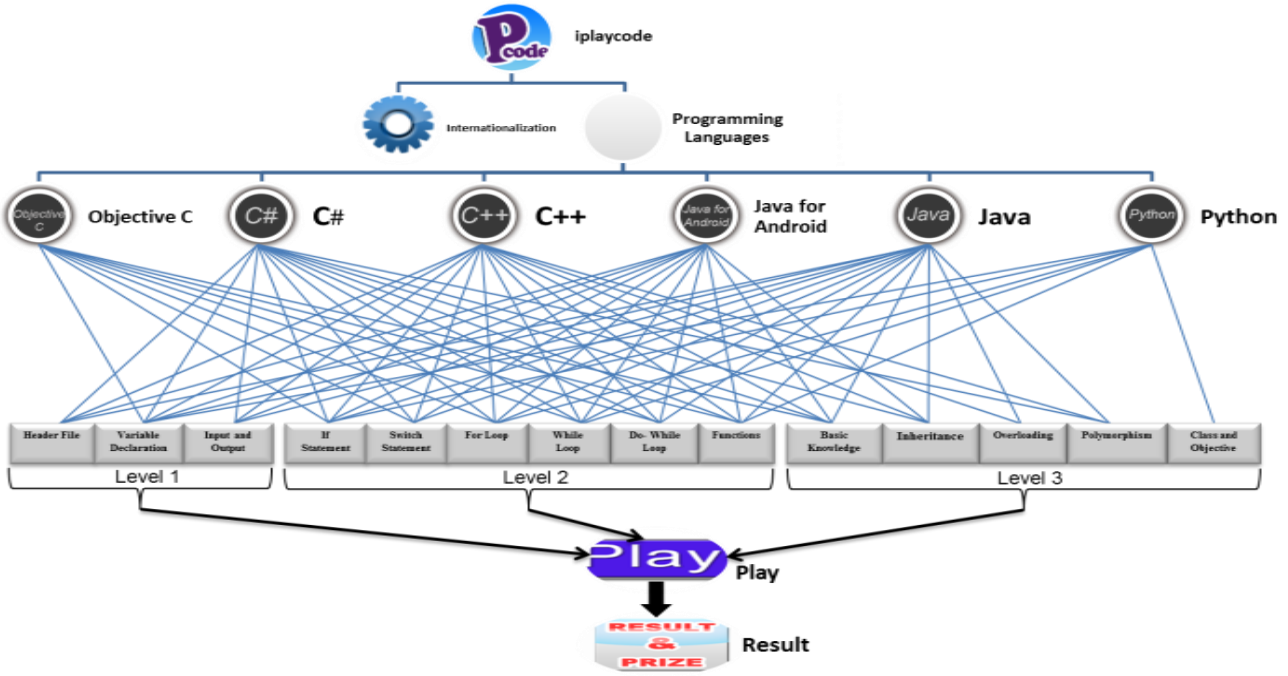


Figure 3. The architecture of iPlayCode.

Polish can be selected using the internationalisation option of the iPlayCode application. The user enters his or her name and a programming language of the user's choice is selected. There are six programming languages in the iPlayCode application: Objective C, C#, C++, Java for Android, Java and Python (Figure 3). Each programming language has three levels but the sub-function in each level differs. The following screenshots (Figures 4 to 10) show the game interface and display for the iPlayCode application. When the game loads, the start screen is displayed. As shown in Figure 4, the players start the game by pressing the 'Play' button after entering their username. The user can then press the 'Settings' button to change the language for the interface if his or her preferred language is not English.

Figure 4. The start screen.

The interface of the iPlayCode after pressing the 'Play' button is shown in Figure 5. The game starts at the main game environment, and players either select the programming language they want to study by pressing one of the buttons or return to the start screen to change the username or language for the user interface.

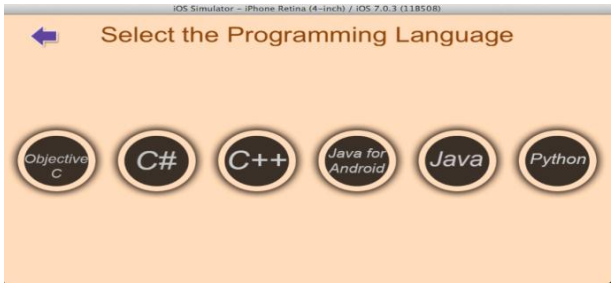
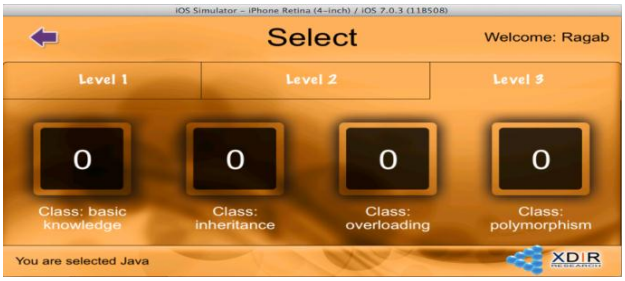


Figure 5. The main game environment.

The different levels of the gameplay process are shown in Figure 6. Figure 6(a) is level 1 and it is made up of one to three sub-functions in each programming language. Figure 6(b) is level 2 and it consists of five to six sub-functions. Figure 6(c) is level 3 and it comprises of one to four sub-functions. In addition, each level screen displays the username, a button to return the main screen and the chosen programming language.

(a)

(b)



(c)

Figure 6. The game screens: (a) level 1, (b) level 2 and (c) level 3.

The questions screen includes the score, the number of questions, the back button to display levels, a timer and two buttons to select whether it is either the correct or the wrong answer (see Figure 7).

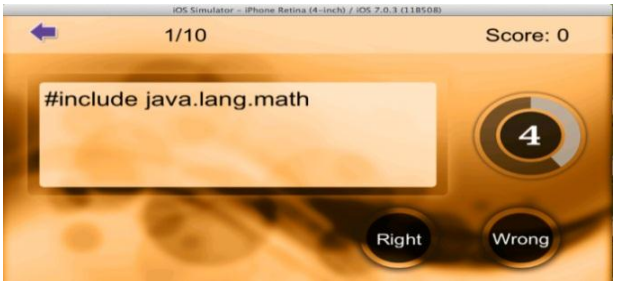


Figure 7. The questions screen

The answers screen (see Figure 8) displays the total score, the number of questions, the back button to display levels, the score for each question and an additional two buttons: one to move to the next question and the other for help to find out the correct answer.

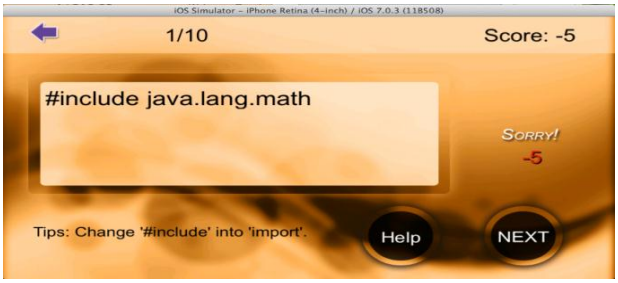


Figure 8. The answers and aid screen

The first screen to display the points for the sub-function played is shown in Figure 9. This screen also includes buttons to return to the sub-function to play again, to choose another level, to go to the final result screen, and a return button to go to the main screen to choose another programming language. In addition, the screen displays the points obtained using a medal graphic.

The type of medal is determined as follows:

- Bronze, if score>0 &&score<40

- Silver if score>39 &&score<70
- Gold if score>69 &&score<101

Figure 9. The first screen to display the points for the sub-function played.

The final results screen for the three levels displays the username (see Figure 10). It also includes the return button and displays the points obtained alongside the corresponding level and medal. The total points and medals are calculated as follows:

Total score = Total score for all sub-functions in the level *(1)*

In the final result screen, the type of medal awarded to any level achieved is obtained using:

$$\frac{\text{Total score for all sub-functions in the level}}{\text{The number of sub-functions in the level}} \quad (2)$$

The award criteria are:

Bronze, if score>0 &&score<40; silver, if score>39 &&score<70; and gold, if score>69 &&score<101

For instance, in Figure 10, level 1 has a total score of 108 and the number of sub-functions is three, thereby awarding a score of 36, which falls within the bronze range. Level 2 has a total score of 345 and the number of sub-functions is six, hence awarding a score of 57.5, which falls within the silver range. Level 3 has a total score of 340 and the number of sub-functions is four. This gives a score of 85, which falls within the gold range.

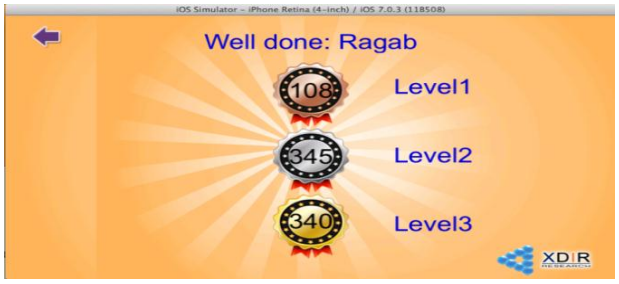


Figure 10. The final result screen of the three levels

V. TESTING AND EVALUATION

Functionality testing was used to trial the game. Game testing is associated with the functionality of the game design and involves playing the game to ascertain whether the game is functioning as specified. The main aim of the functionality test is to unearth any general problems within the design or user interface. The application was tested using the iOS 7.0.3 (11B508) simulator.

Subsequently, the application was deployed in mobile devices including the iPhone and iPad. The results of the tests prove that the game is free of any defects since the application operated smoothly without any errors or breakdowns. The game mechanics were also tested to ensure that it worked properly on several mobile devices.

To ensure consistency the test has been repeated. The results obtained provide further proof that the game functions without errors and faults. The asset of utmost importance, however, is the iPlayCode, which ensures the integrity of the application. Table (1) shows the results of the test, which confirms that devices like the iPhone and iPad are able to support the iPlayCode.

TABLE I. THE TEST THAT DEVICES LIKE IPHONE AND IPAD ARE ABLE TO SUPPORT THE IPLAYCODE

Functions	Work	Not work	Description
Login function	Yes		Successful
Menus functions	Yes		Successful
Button 'Setting' in starting screen	Yes		Successful
Button 'Play' in starting screen	Yes		Successful
Button 'Return' in all screens	Yes		Successful
Buttons 'Right, Wrong, Help, End' in answering screen	Yes		Successful
Buttons 'MENU, RESULT & PRIZE'' in the first result screen	Yes		Successful
Timer in answering screen	Yes		Successful
Labels in all screens	Yes		Successful
Scroll label in answering screen	Yes		Successful
sound	Yes		Successful
Background	Yes		Successful

A detailed breakdown of the different tests is shown in table (1), which includes: functional testing, performance testing and usability testing. These results show that there are no problems with the functionality of the application.

VI. DISCUSSION

iPlayCode exists to resolve the problems caused by the conventional, rigid method of teaching utilised by tertiary institutions. To be effective, a game-based learning platform must emphasise the learning objectives and provide students with a resource that is both entertaining and supportive. This goal is fulfilled through clear and simple user interface development that helps students to learn programming languages. The user element is the core feature of the application, particularly games.

Three factors need to be considered when creating a user interface: beginning the game, basic mechanics and ease of use.

A. *Beginning the Game*

The start of the game is an important stage for the user. If the early stages are confusing and difficult it impedes the user's enjoyment, causes frustration and may result in the user quitting. iPlayCode begins with an enticing and clear interface that simply asks the user to specify their preferred language and enter a username (see Figure 3). The evaluation of the platform shows that it incorporates most of the essential aspects of what is deemed a 'good game'. The objectives and regulations are straightforward and this creates a product that is simple and enjoyable. It provides students with a learning tool that is relevant, practical and interactive. The interactive experience is enhanced by the iPlayCode application, which gives the students tasks that are relevant to the acquisition of valuable information.

 ISBN: 978-1-61208-416-9

B. Game Mechanics

Entertainment and learning are influenced by the game's mechanics. Two key features are incorporated into iPlayCode: incentives and competition.

When students interact with games, positive competition is important. iPlayCode encourages the completion of tasks by prompting the students to replay each level until they obtain a high score (see Figure 8).

This repetition ensures that the students easily recall the information and solve the problems. The students are encouraged and engaged by the penalties and rewards contained within the application. The game offers gold, silver and bronze medal achievements that are awarded when the students earn a certain number of points for each level (see Figure 9). A penalty of minus five points is given if a student answers incorrectly, does not give an answer within the allotted time, or fails to answer the question correctly before the time is up (see Figure 7).

C. Ease of Use

The enjoyment of the game is closely related to its usability. A game that is difficult to operate frustrates the user, which means they do not take pleasure in using the application. Figures 5, 6 and 7 illustrate that the simple iPlayCode user interface decreases user processes, appeals to students and guarantees fun.

VII. Conclusion and Future Work

This study presents the development of the game-based learning platform iPlayCode, which offers university students a new and exciting method of learning programming languages. Many existing educational games developed for programming languages are restrictive, and they often do not cover the vocabulary of this process in its entirety. The iPlayCode platform has conquered this disadvantage by incorporating six different programming languages: Objective C, C#, C++, Java for Android, Java and Python. Moreover, the application screens are simple and easy to use. The conclusions drawn from this paper are:

1) A good user interface design that is easy to use makes an application attractive and interactive; as a result, users' learning needs are more effectively met.
2) Game-based learning creates a viable educational environment that further promotes interactive learning.
3) Game-based learning improves the efficiency of learning and makes it engaging and fun.

It is hoped that the aspects of the game discussed in this research paper will be implemented and tested by different users, groups and individuals, and potentially develop to include other programming languages. The long-term aim is that the platform will reach increasingly varied groups of learners. The findings gathered from this study can also be used to pinpoint even more valuable conclusions related to the worth of mobile games and the ways in which they can be further refined.

References

[1] A. Herrington and J. Herrington, "Authentic mobile learning in higher education," presented at the AARE 2007 International Educational Research Conference, Fremantle, Western Australia, 2007, November 28, pp. 1-10.

[2] M. Milrad, "How should learning activities using mobile technologies be designed to support innovative educational practices?," Big issues in mobile learning, University of Nottingham, Nottingham, 2006, pp. 27-29.

[3] N. Winters, "What is mobile learning," Big issues in mobile learning, University of Nottingham, Nottingham, 2006, pp. 4-7.

[4] C.-C. Chao, "An investigation of learning style differences and attitudes toward digital game-based learning among mobile users," in Wireless, Mobile and Ubiquitous Technology in Education, 2006. WMUTE'06. Fourth IEEE International Workshop on, Athens, Greece, 2006, November 16-17, ISBN: 0-7695-2723, pp. 29-31.

[5] M. O. M. El-Hussein and J. C. Cronje, "Defining Mobile Learning in the Higher Education Landscape," Educational Technology & Society, vol. 13, pp. 12-21, 2010, October, ISSN: 1436-4522, PP. 12-21.

[6] S. H. Ab Hamid and L. Y. Fung, "Learn programming by using mobile edutainment game approach," in Digital Game and Intelligent Toy Enhanced Learning, 2007. DIGITEL'07. The First IEEE International Workshop on, Jhongli, Taiwan 2007, March 26-28, ISBN: 0-7695-2801-5, pp. 170-172.

[7] J. Bowen and S. Reeves, "Refinement for user interface designs," Electronic Notes in Theoretical Computer Science, Proceedings of the 2nd International Workshop on Formal Methods for Interactive Systems (FMIS 2007), Hamilton, New Zealand, vol. 208, 2008, April 14, ISSN: 1571-0661, pp. 5-22.

[8] V. Kadyte, "Learning can happen anywhere: a mobile system for language learning," Learning with mobile devices, Learning and Skills Development Agency, London, UK, 2004, ISBN: 1-85338-833-5, pp. 73-78.

[9] J J.-H. Valk, A. T. Rashid, and L. Elder, "Using mobile phones to improve educational outcomes: An analysis of evidence from Asia," The International Review of Research in Open and Distance Learning, Edmonton, Canada, vol. 11, 2010, March, ISSN: 1492-3831, pp. 117-140.

[10] T.-H. Tan and T.-Y. Liu, "The mobile-based interactive learning environment (MOBILE) and a case study for assisting elementary school English learning," in Advanced Learning Technologies (ICALT04), 2004. Proceedings. IEEE International Conference on, Washington, USA, 2004, 30 Aug.-1 Sept, ISBN: 0-7695-2181-9, pp. 530-534.

[11] C. Ware, Visual thinking: For design: Morgan Kaufmann, 2010.

[12] J. Zhang and J. Lu, "Using Mobile Serious Games for Learning Programming," in the proceedings of The Fourth International Conference on Advanced Communications and Computation (INFOCOMP 2014), Paris, France, 2014, July 20-24, ISBN: 978-1-61208-365-0, pp. 24-29.

[13] R. Ihnissi and J. Lu, "An investigation into the problems of user oriented interfaces in mobile applications," presented at the The 2014 World Congress in Computer Science, Computer Engineering, and Applied Computing (WORLDCOMP'14), Las Vegas, USA, 2014, July 21-25, pp. 1-7.

[14] T. W. Malone, "What makes things fun to learn? Heuristics for designing instructional computer games," in Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems, New York, USA, 1980, ISBN: 0-89791-024-9 pp. 162-169.

 ISBN: 978-1-61208-416-9