Educational MMORPG for Computer Science: DeBugger, a Virtual Lab on PC and Smart Phones

Ilmi Yoon[1], Arno Puder[1], Gary Ng[1], Manori Thakur[1],
Hunvil Rodrigues[1], and Jae H. Paik[2]
[1]Computer Science, SFSU, [2]Psychology, SFSU
San Francisco, CA, USA

Eun-Young Kang
Computer Science, CSULA
Los Angeles, CA, USA

***Abstract*— There are two important observed facts for this research. One is that computer games with social interactions have become strong cultures to young students. The other is that introducing a computer programming language to majority of students is challenging as programming paradigm and concepts are completely alien to most of them. In this research, we introduce a Multiplayer Online Role Playing Game (MMORPG) named "DeBugger". It is developed to utilize the MMORPG online culture to introduce computer science education to improve learning outcome. Considering the increase in awareness and early exposure to programming at high school or even at middle school, developing publicly available multiplayer online educational games for CS students is a timely and sought effort. The DeBugger aims to build a collection of educational mini games within a virtual community of learners where players fight with bugs by solving problems, interact with other players and continue to play on smart phones during mobile gaming. The DeBugger is designed to take advantage of social interactions, mobile games and community to retain players longer, promote players to solve more quests, and encourage players to discuss and learn from each other more actively, which encourages peer learning and tutoring. A positive efficacy testing of the DeBugger on learning and social interaction among students will be discussed in paper.**

Keywords— Multiplayer Online Role Playing Game; Video Game; Java; Programming; Computer Education; Mobile Games

I. INTRODUCTION

Recent computer science education strongly recognized the significance of abundant exposure at an early age in improving one's computational thinking and problem solving approach [1,2]. Similar to Algebra, for example, where concept of variables in algebraic expressions takes some exposure time to get familiar to most 5-7th grade students, concept of variables in programming language also takes some time to become familiar. Moreover, learning computational approach usually takes sufficiently iterative practices for core concepts (variables, data types, logical flow, conditional statements, iteration, objects, methods, etc…) for most students whether young or not. To shorten the learning curve and to entertain the iterative learning process, game-based learning tools can be an effective aid for computer education.

MOOC (Massive Open Online Course) is one of noticeable approach in education. Among many computer science courses, introductory computer science course (CS1) has become pilot course for MOOC and diverse approaches are being explored. Majority of these approaches focuses on lecturing mechanism and interactions between the instructor and students even though peer interactions have served important roles for learning explicitly or implicitly [3,4,5]. As much as effective instructor-student interactions are explored for MOOC, similar effort should be made to explore and develop means to improve peer interactions. Here, we argue that virtual community where students can meet each other and play collaboratively or competitively via educational game challenges will serve as virtual labs and provide an atmosphere for desirable peer interaction. In addition, DeBugger game can be continuously played on Android and iPhone while students are away from computer.

Section 2 discusses related work; using multiplayer game for educational purposes and other game-like approach for CS education. Section 3 discusses the principles of designing the DeBugger game and section 4 discusses implementations of Android version and cross-compilation of the game for iPhone version, community features within the DeBugger game, and game architect. Section 5 presents results of user trial and on-going efficacy evaluation and implementations. Section 6 concludes the paper.

II. RELATED WORK

Game-based learning has been an active area of educational research with hopes that playing game would effectively aid students' learning. Large number of studies showed the educational impact of the game-based approach [6].

Educational games (also called edutainment, serious game, gamification) have been adapted and used in diverse areas such as elementary math to professional medical trainings [7,8]. Recent study shows that game-based education reaches beyond original objective of educational and motivates players to achieve much more [9]. Cooper [9] utilized multiplayer online game as social interactions serve well to motivate players. Other studies also showed significance of social interactions in the game to retain players longer. Figure 1 shows the general play time of stand-alone game vs. MMORPG game; games with social interactions keep players longer over time [10]. We observed that popular MMORPG games are retaining their players for very long period time (months to years) and invite friends to play together (spread through human network). Recently, Farmville, a relatively simple farm nurturing (crops or farm animals) Facebook game made a nice show case of how quickly a game can spread through social interactions (80 million active players) and keep players playing repeatedly.

Hours per week	MMORPG players	Non-MMORPG players
0–1 h	1%	11%
1–2 h	5%	38%
3–6 h	10%	35%
7–10 h	13%	7%
11–20 h	25%	2%
21–40 h	34%	4%
40 or more h	11%	2%

Figure 1. MMORPG vs. non-MMORPG games, hours played; Source: Addiction to the Internet and Online Gaming.

Popular MMORPG games like World of Warcraft has several millions players. Lots of research has been done to analyze the main driving factors [11,12]. These studies pointed out that players especially enjoyed the fact that there is a community of audiences to give compliments when player achieved high level or completed a quest. It is an important finding which can significantly increase educational games' efficacy. Educational games that are well designed for good educational impact often suffer from the fact that players do not play voluntarily over and over. Another strong trend of young generation is continuous usage of smart phone at all times. Mobile DeBugger game is created to take advantage of this mobile trend, to encourage students to connect to other players, and seamlessly continue playing DeBugger game while students are away from computer.

The MMORPG game, "DeBugger" was designed and developed to make use of these findings to assist Introductory Computer Science course. It is widely accepted that programming is very difficult to learn[13]. For example Bergin and Reilly noted that it is well known in the Computer Science Education (CSE) community that students have difficulty with programming courses which can lead to high drop-out and failure rates [14]. The concepts in CSE are non-intuitive or/and overwhelming. Students need lots of exercises to digest the concepts, with close interactions, guidance and help when they are lost. Average pass rate for CSE course at colleges across US is around 60% [15]. Considering that significant shortage of workforce in CS [16], it is imperative to develop and provide publicly available educational game for CS students to assist students as well as instructors.

III. DeBugger Game

The DeBugger is a MMORPG game where there is a persistent virtual world that continues to exist and progress even after a player exits the world (i.e. stop playing the game). In that world, players are represented by their characters with status properties like game level, virtual money, health, list of friends, game items like weapons or tools.

Design principles of the DeBugger game are; (1) Make use of players attachment to the character. In commercial MMORPG games, players have shown strong attachments to their online status (or their own characters). To be able to level up their characters, players can enjoy irksome process –known as boring, time consuming and repetitive tasks [17]. Players happily spend hours and hours to succeed in a quest. While there is pleasure of succeeding in quests itself, players are strongly motivated by the fact that their successes in quest result in level-up or acquisition of special awards and recognition in the community. Therefore, to level up characters in DeBugger, players are often observed to play DeBugger longer with focus to keep their rank high. (2) Studies have shown that students learn from peers as much as from teachers [18,19,20,21]. Students tend to ask questions for clarifications among peers first and then ask questions to teacher if no answer can be found among peers. In addition, peer pressure pushes students not to be left behind. Online game communities have shown that experienced players take great pleasure helping novice players. These advanced players can be further motivated to help new players like a TA giving close individual interactions that teachers may not give in real world. Frustration from the learning can be alleviated by sharing the similar troubles of individual with peers. The peers can be connected in the virtual world, available when needed, and increase educational impact. (3) For timid students, they can hide behind their virtual character, to avoid the fear of failure. (4) Importantly, typical MMORPG games advance and adjust depending on the needs or demands of the player community unlike console games that have pre-fixed settings once manufactured. The DeBugger game can advance and adjust the same way. In addition, the DeBugger can collect through the game server, a massive amount of educationally meaningful data based on the players' activities. These data can be useful for iterative design cycle (design, development, user test) of the DeBugger game for various users to maximize their learning outcome.

One thing to note is that the DeBugger game is not intended to replace traditional classes, but to be used as an effective support tool like a virtual lab with a TA that provides repetitive practices (lab) and feedback and clarifications from other players (virtual TA), so students can master the core pedagogical components to then progress smoothly.

In the DeBugger game, there are many mini games that players can play with other players or against bugs; the DeBugger title was inspired by the origin of the word (removing a bug from computer to fix errors). These mini games (games inside of DeBugger game) are intended for players to develop competence in computer science concepts and fundamental computational approaches when played repeatedly. In the DeBugger game, players 1) fight with bugs by solving flashcard games - chapter review questions after reading text book, 2) play "CodeGame" to solve a given mission by dragging and dropping the proper code segments in correct sequence, 3) play "Variable" games to practice the variables, data types and operations in Tetris style game, 4) play board game with other players by solving quiz style questions, and 5) interact with other players via chatting, friends list, reputation list and other features.

IV. Implementation

A. Mini Games

Students who take first Computer Programming courses are introduced to the core concepts of problem solving with pseudo-code, variables, data types and operators, flow controls (selection and iteration), methods, strings, and arrays in

sequential order with slight variations. Each mini game has a specific focus so it is designed to work for one concept or one group of a few related concepts.

Figure 2. "CodeGame" Interface: a mission is displayed at upper left corner and the answer segments to drag are displayed below the mission. A segment can be dragged and dropped individually.

Figure 3. "Variable" game. Player is supposed to fill in the left column where variable slots are, according to instructions that appear one at one time on the right column.

The mini game called "CodeGame" is for learning and practicing program structure, importance of syntax rules, and simple program flow. When the mini game starts, a player will receive a simple mission that is randomly assigned according to the player's current level with a list of possible answer segments displayed on the left panel as shown in figure 2. To solve the mission, the player should pick up correct answer segments and then arrange them on the right answer panel in proper sequence by drag-and-drop. The goal of this mini game is to enable the player to comprehend the functionality of given answer segments and then focus on the program logic. The answer segments can be either actual program codes or abstract pseudo code as shown in figure 2. Abstract high level code helps students learn problem solving using top-down or divide-conquer approach without thinking of the syntax details. As for the actual program code answers, the mini game presents both grammatically correct and incorrect answers. In this case, to solve the mission, the player must know the precise language syntax and this allows the player to review important language syntax. Screen capture of playing this mini game is recorded and available on YouTube (http://www.youtube.com/watch?v=wfFW7mfd1M4).

"Variable" game is designed to teach concepts of variable, data type and operations associated with data type. For example, the minigame shows instructions related to a variable 'i' type of int, such as i++, then the player should calculate the value for the variable and enter the value to a variable slot for 'i' on the left column within a given time. An instruction can involve more than one variable such as j = i/3. In this case, the player should read value from i slot and perform integer division. When answer is wrong or the player has reached the time limit, then the row turns to red and stays red, while the right answer turns the line green and clears out. Wrong answers stack up in red rows and the player loses when the stack reaches the top (see the figure 3). Also, the screen capture of this mini game being played is recorded and available on YouTube (http://www.youtube.com/watch?v=vtcVQG-n8e0).

Figure 4a (top) & 4b (bottom). A player is fighting with a bug. Bug asks a question and the player has to answer. Health decreases as time lapses. Player's health, level, and gold are shown at the upper left corner. Lower left corner panel serves as a message panel and chat panel.

Based on the designed game architecture, many more mini games can be easily added to allow players to be exposed to further concepts continuously with fun (Figure 4a & 4b).

Players can seamlessly continue to play the game while they are away from the PC. Mobile DeBugger game allows for mini games play and maintain the player's inventory, score and all other belongings since they share the same account as the game server and DB server. Currently players can solve multiple choice questions that enable collection of stones for correct answers, so players can directly fight with bugs with stones as weapon. Mobile DeBugger makes use of touch pad for throwing stones in natural pattern, GPS information for finding friends nearby, and accelerometer for shaking off bugs or wrong options during the play.

Figure 5. Screen capture of Mobile Debugger. Players can throw stones to the bugs using the touch pad screen,

B. *Social Activity*

There are many social activity features within the DeBugger game. Players can chat with other players in a few different ways. Public chat is for chatting with anyone in the same virtual space (room) and private chat is for chatting with a specific person that a player wants to chat with. Also, it is possible to chat with friends who are online. A player can manage a friends list. Player can choose to show their level and other performance to their friends or the public

Player's achievement can be measured by levels, number of questions played and the accuracy ratio. Achievement board displays the top 10 high score achievers to encourage players to play longer and better (Figure 6).

Figure 6. – Player's performance is measured in diverse ways, and players can see other players' performance unless blocked by the player.

Social gaming and MMORPG are relatively recent game genres as they gain popularity over the availability of Internet. Social interactions within the game generates excitement of the game which allows popular games to easily develop communities of multimillion players. Once community of learners is created, students can learn not only from educational game components but also from their peers through discussion of problems with them. The DeBugger game was created with the vision of nurturing such a community of learners playing collection of games together, helping each other and inviting more friends to join.

The DeBugger is not only for students. The tool assesses students' learning outcomes automatically and provides useful information (e.g. grading of homework) back to teacher. This is very useful and efficient in that teachers save time for grading and students receive feedback immediately. Figure 7 shows the data available for teachers.

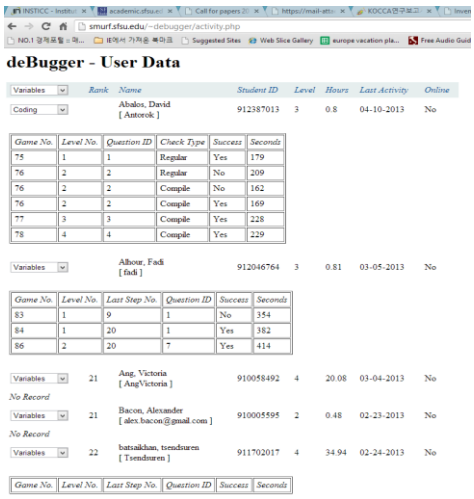

Figure 7. Website where teachers can see all of the students' performance in DeBugger.

C. *Game Architecture*

For DeBugger game, we need to create and support a virtual world that exists continuously on the game server. The game server runs at all time, is used by all clients to play the game, and updates the Database Server. A game client establishes a connection to the game server when a player wants to play the game, receive the latest status of the virtual world that has changed since the last time player logged out, and allows the player to interact with other players who are online. The database server keeps all the data of individual players (health, level, money, game items, friend's list, performance, and etc) and other data to maintain the game world continuously. An efficient game protocol between the server and clients is developed to make the communication effective. The DeBugger game also has a bug server that maintains all the bug characters – how often they appear (spawn), what kind of game item they drop, how aggressively attack players, etc. The bug server also controls bugs' behaviors, such as wandering and reacting to collisions, with simple AI (Artificial Intelligence) to improve the player's fun experience with the game.

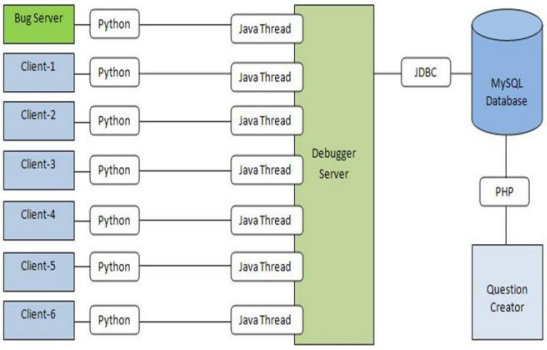

Figure 8. DeBugger Architect. It shows the relationship between a Client and the Server

Figure 8 shows an architect of the DeBugger game, depicting each component and their connections to each other. The game server was developed using JAVA and MySQL as DB server. Bug server was extended from game client that already included collision handling. The game client was developed utilizing Panda3D, an open source game engine, and python scripts. Figure 9 shows the further development of mobile DeBugger connecting game server directly, supporting seamless game play for players. We also employ XMLVM cross compilation technology for effortless porting from Android to iPhone version.

Figure 9. DeBugger Architect including Mobile Debugger. This shows the relationship between a PC Client, Android Client, iPhone Client and the Server

Stable support of the DeBugger game server is critically important for MMORPG game. We evaluated the scalability utilizing Clouds Computing acting as individual clients and the DeBugger server can stably support 100 concurrent players at more than 100 heartbeat per second performance. Figure 10 shows a screen capture showing all the clients running in the Amazon Clouds in a same room [22].

Figure 10. Testing Server Scalability using Amazon Clouds.

V. RESULTS

A. Project Efficacy Testing

The DeBugger game has been used at SFSU CSc 210 Introduction to Programming using JAVA during Spring 2012 (pilot test) & Spring 2013 (official efficacy testing). Usability and learning impact of the DeBugger is continuously being evaluated. During the Spring 2013, 29 student participants were recruited from 4 introductory Computer Science programming classes. Seventeen students were introduced to DeBugger and were asked to play at the minimum 1 hour a week for the first 2 months as a part of their learning (Experimental Group) while 12 students were not given such exposure to DeBugger (Control Group). All students were given a pre-test that examined their general knowledge about JAVA programming (e.g., Data types, Variables, Syntax flow) in order to set the baseline performance scores. Mixture of forced choice (i.e., true/false and multiple choice) and open-ended problem solving questions were included. Once pre-test was completed, students in the Experimental Group were given the access to DeBugger for the next 8 weeks. To examine the effectiveness of playing DeBugger while taking CSc 210, all students were also given a post-test at the end of the study. The questions on the post-test were identical to the pre-test and students' improved scores from pre- and post-test across two groups were compared. We are currently in the process of analyzing the data. However, our preliminary data analyses are already showing positive impact of playing DeBugger on student learning outcomes.

Figure 11 illustrates participants' mean percent correct on the pre- and post-tests. It was found that overall, students' general knowledge about JAVA programming improved significantly from pre-test to post-test, $F(1,27) = 7.49$, $p = 0.01$, $\eta 2=0.22$. However, such significant improvement was driven mostly by the Experimental Group as alluded by the significant interaction between the group and testing phase, $F(1,27) = 4.6$, $p = 0.04$, $\eta 2=0.15$. Indeed, follow-up post-hoc analysis confirmed that students' performance improved significantly from pre- to post-test only in the Experimental Group where DeBugger was integrated into their learning experience (pre-test: M = 44.12; SD = 13.25; post-test: M = 64.71; SD = 16.25), $t(16) = 4.42$, $p < 0.001$. For the Control Group, there was no significant improvement in students' scores from pre- to post-test (pre-test: M = 54.17; SD = 11.65; post-test: M = 56.68; SD = 20.6), $t(11) = 0.33$.

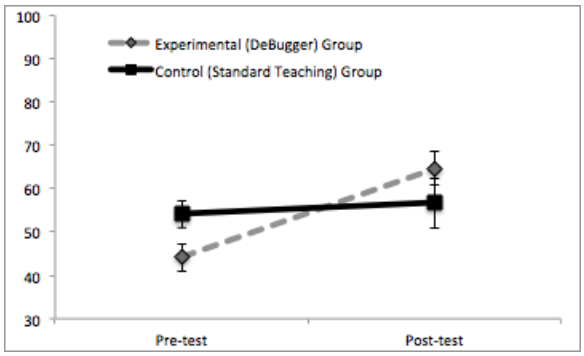

Figure 11. Students' Mean Percentage Scores on Pre- and Post-tests.

B. Student Learning Experience

The ffollowing comments from students in the Experimental Group provide noteworthy social cooperative and interactive nature of the DeBugger experience that seems

highly conducive for acquiring knowledge while enjoying the process of learning:

"We played the board game for pretty much the whole hour. It was fun. I explained as much as I could to John, and we had a great time. Debugger is a great place to talk about coding and Java and just hang out. "

"There were two other players from CSc 210 that played Debugger today. We played the bugs and board game for about 30 - 40 minutes. It was really fun, and we talked a lot about programming and school and stuff. When we both played the board game at level three, and it's more fun at a harder level because both players can talk about the problem and help each other out with the solution."

"I was able to explain to the other player about short circuiting, compile/run time errors, and other stuff. It was time well spent. I really enjoyed playing Debugger today."

Based on the present research findings, we advocate for educational studies that thoroughly investigates the learning impact of individual mini games for pedagogical practices and consequential student learning experience (e.g., peer interactions in virtual community).

C. On-Going Development

Further developments are in progress. First, additional mini game to add is one for practicing selection flow control concept. It will utilize idea of racing game and player's route will be selected depending on the value in the car and the condition at the branch. To be able to finish the lap in given time, player should choose the value wisely, to utilize the conditions at the branch, and to stay at desired routes. This game will be extended to repetition flow control by running multiple laps. For the beginners, the car move very slowly to give enough response time and the car will accelerate as player progresses well. Second game to add is for practicing repetition flow control, specifically nested repetition flow control using pattern printing style arcade game. The third addition is a game for practicing array and repetition flow control.

VI. CONCLUSION

We introduced a MMORPG game, "DeBugger", that can be used as a primary assistance tool to improve students' learning outcome for Introductory Computer Science course through inherent properties of the MMORPG: addictive and fun game play for repetitive activities and social interaction which results in longer retention and active peer interaction/tutoring. The evaluation study shows that the impact of the game is effective and socially cooperative as a virtual lab and space for peer instruction.

ACKNOWLEDGMENT

This project is being partially funded by National Science Foundation NSF DBI- 0543614 and NSF TUES 1140939. CSc 631/831 Multiplayer Game Design and Development course students at fall 2009 contributed to majority of game concept, art work, DB, game contents, client and server implementation.

REFERENCES

[1] V. Barr and C. Stephenson, Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community?, ACM Inroads, Vol. 2 No. 1, March 2011.

[2] Lee, Irene, et al., Computational Thinking for Youth in Practice, ACM Inroads, Vol. 2, No. 1, March 2011.

[3] C. Greenwood, J. Delquadri, and R. V. Hall, Longitudinal effects of classwide peer tutoring, Journal of Educational Psychology, Vol 81(3), Sep 1989, 371-383.

[4] A. King, A. Staffieri, and A. Adelgais, Mutual peer tutoring: Effects of structuring tutorial interaction to scaffold peer learning, Journal of Educational Psychology, Vol 90(1), Mar 1998.

[5] A. Kolb, The evolution of a conversational learning space, In A. C. Baker, et al. (Eds.) Conversational learning: An experimental approach to knowledge creation. Westport, CT: Quorum Books, 2002.

[6] Papstergiou, M., Digital Game-Based Learning in High School Computer Science Education: Impact on Educational Effectiveness and Student Motivation, Computers & Education, v52 n1 p1-12 Jan 2009.

[7] GREEN, C S, AND BAVELIER, D. 2003. Action video game modifies visual selective attention. Nature 423 (2003), 534-537, Letters to Nature.

[8] PRENSKY, M. 2001. Digital Game-Based Learning. McGraw-Hill, New York. ISBN 0-07-136344-0.

[9] Cooper, et al, Predicting protein structures with a multiplayer online game, Nature 466, pp 756-760, August, 2010.

[10] Ng, Brian D. and Peter Wiemer-Hastings, Addiction to the Internet and Online Gaming, Cyber Psychology & Behavior 8.2 (2005)

[11] Seungkeun Song, Joohyeon Lee (2007), Key factors of heuristic evaluation for game design: Towards massively multi-player online role-playing game, International Journal of Human-Computer Studies, no. 65, 709-723

[12] Yee, N. (2002) Understanding MMORPG Addiction

[13] T. Jenkins. On the Difficulty of Learning tp Program. In Proceedings for the 3rd Annual conference of the LTSN Centre for Information and Computer Sciences , Loughborough, UK August 27 - 29, 2002.

[14] S. Bergin and R. Reilly. The influence of motivation and comfort-level on learning to program. In Proceedings of the 17th Annual Workshop of the Psychology of Programming Interest Group pages 293-304, University of Sussex, Brighton UK 2005.

[15] Jens Bennedsen and Michael E. Caspersen, Failure rates in introductory programming,
SIGCSE 2007, pp 32-36.

[16] Computing Education for 21st Century (CE21) Program Solicitation NSF 12-527 http://www.nsf.gov/pubs/2012/nsf12527/nsf12527.htm

[17] S. H. Hsu, , M. H. Wen, and M. Wu, Exploring user experiences as predictors of MMORPG addiction, Elsevier, Computers & Education, Vol. 53, No. 3, pp. 990–999, November 2009.

[18] Christudason, A. Successful Peer Learning, http://www.cdtl.nus.edu.sg/success/sl37.htm

[19] Falchikov, N., Blythman, M., Learning together: peer tutoring in higher education.

[20] Johnson, D.W.; Johnson, R.T.; & Holubec, E.J. (1993). Circles of Learning. Edina, MI: Interaction Book Company.

[21] Kaufman, D.B.; Felder, R.M.; & Fuller, H. 'Peer Ratings in Cooperative Learning Teams'. Proceedings of the 1999 Annual ASEE Meeting, ASEE, Session 1430'.

[22] Athalye, A., "Extending the DeBugger Game by Adding 3D Models and Testing Server Scalability" SFSU CS Master Project, 2011, http://cs.sfsu.edu/techreports/11/CE-11.07.htm