DeBugger Game: Mobile Virtual Lab for Introductory Computer Programming Courses

Ilmi Yoon, Eun-Young Kang, Oh-Young Kwon
Computer Science, San Francisco State University/ Computer Science, California State University, Los Angeles/ Computer Science and Engineering, Korea Tech

Abstract

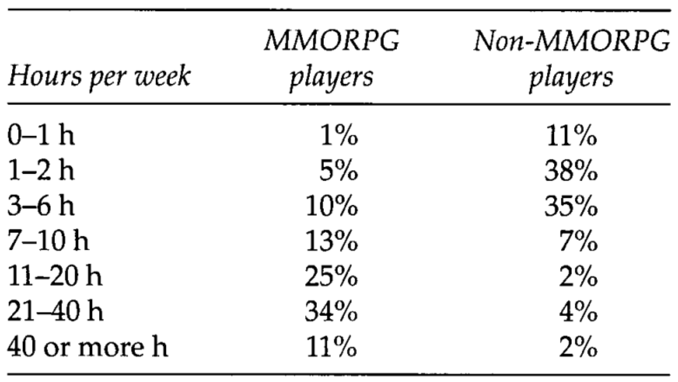
This paper introduces a Multiplayer Online Role Playing Game (MMORPG) named *DeBugger*. The game is developed based on several significant findings about MMORPG games in order to teach introductory Computer Science programming more effectively and provide more exciting learning experience to students. The DeBugger provides a collection of educational mini games within a virtual community of learners where players engage in fighting with bugs, defeat them by solving programming problems, keep track of their scores, manage their characters, and interact with other players to build relationships. The game is implemented to utilize addictive and compelling aspects of MMORPGs such as scoring system, beating the game, role-playing, and online relationship, to retain players longer, promote players to solve more quests, and encourage players to discuss and learn from each other more actively. The game allows the instructor to create a virtual lab to promote peer learning and tutoring. In a virtual lab, players can encounter peers who can teach each other or TAs who can provide more guided help. The game can be run on mobile devices as well as a desktop computer so that students can continue to play the game at their convenience and painlessly increase learning hours. The game is designed to be used as a supplementary lab tool rather than a primary teaching tool. Considering that awareness of importance of early programming exposure to high school or even middle school students has increased, developing publicly available multiplayer online educational games for CS students is a seemly and timely sought effort. A positive efficacy testing of the DeBugger on learning and social interaction among students is discussed in the paper.

1. Introduction

It is widely accepted that programming is very difficult to learn[1]. 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[2]. The concepts in CSE are non-intuitive or/and overwhelming. Students need to solve a lot of exercise problems 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%[3]. Considering that significant shortage of workforce in CS[4], it is imperative to develop and provide publicly available and effective educational tool to help students as well as instructors.

Recent computer science education that strongly recognized the significance of abundant exposure at an early age in improving one's computational thinking and problem solving approach[5,6]. 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 and comfortable to college students. Moreover, to develop computational thinking for programming in addition to learning core concepts (variables, data types, conditional statements, etc.), students, whether young or not, usually need to go through many exercises and, to some degree, repetitive practice. To shorten the learning curve and to make the iterative learning process entertaining while increasing exposure time effectively, game-based learning tools can be an effective aid for computer education.

Game-based learning has been an active area of educational research with hopes that playing game would effectively aid students' learning. A large number of studies have shown the educational impact of the game-based approach[7]. Challenges of educational games are not that they are not educational enough, but that they lack of entertainment and fail to retain players enough to deliver the intended educational impact. The good news is that these studies have shown 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[8]. Popular MMORPG games tend to retain their players for a very long period time (months to years) and allow them to 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 showcase 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 by players; Source: Addiction to the Internet and Online Gaming. 70% of MMORPG players spend more than 10 hours per week to play their games while only 16% of Non-MMORPG players spend more than 10 hours.

Popular MMORPG games like World of Warcraft has several millions players. A lot of researches have analyzed the main driving factors[9,10]. These studies pointed out that players especially enjoyed the fact that there is a community of audience to give compliments when a 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 and playing rather simple games for killing time.

MOOC (Massive Open Online Course) is one of noticeable approach in recent education. Among many computer science courses, introductory computer science course (CS1) has become pilot course for MOOC and diverse approaches are being explored. However, 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[11,12,13]. 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.

The MMORPG game, "DeBugger" was designed and developed to make use of these findings to assist Introductory Computer Science course. In the game, players engage in fighting with bugs, defeat them by solving programming problems, keep track of their scores, manage their characters, and interact with other players to build relationships. Also, the game enables the instructor to create a virtual community (virtual lab) where students can meet each other and play collaboratively or competitively via educational game challenges. 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 the computer.

2. 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, and 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 attachment to their online avatar (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[14]. 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[15,16,17,18]. Students tend to ask questions for clarification among peers first and then ask questions to the teacher if no answer can be found among their 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. 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 practice (lab) and feedback and clarifications from other players (virtual TA), so students can master the core pedagogical components to then progress smoothly.

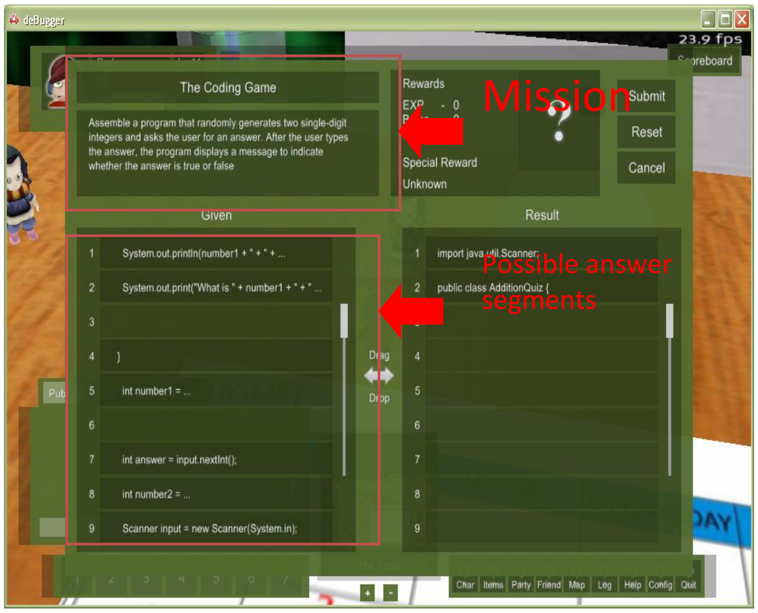
3. DeBugger Implementation

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.

3.1 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.

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 the 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 code or abstract pseudo code as shown in the 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 correct and incorrect answers in grammar. 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).

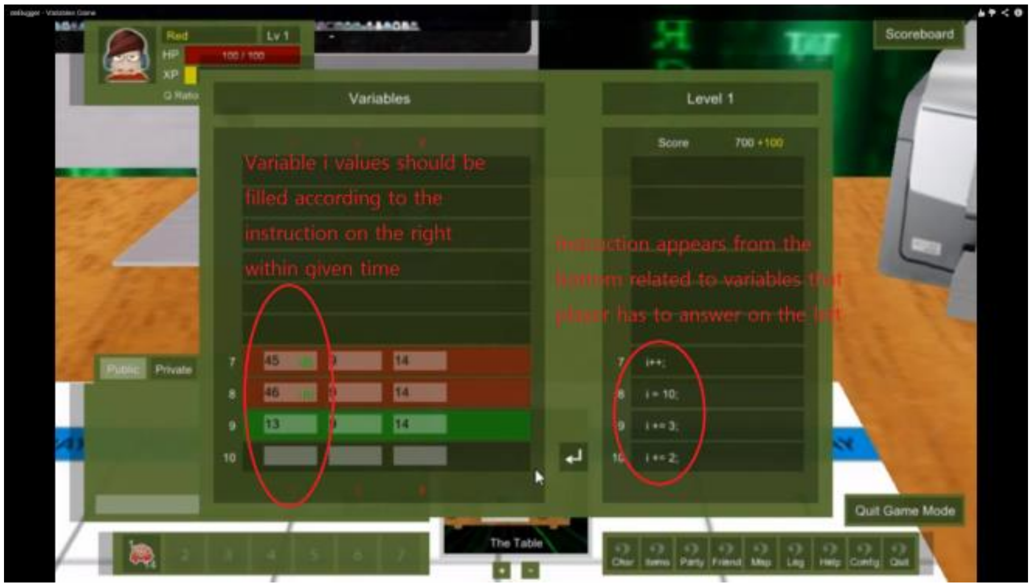


Assemble a program that randomly generates two single-digit integers and asks the user for an answer. After the user types the answer, the program displays a message to indicate whether the answer is true or false

```
1   System.out.println(number1 + " " + " + ...
2   System.out.print("What is " + number1 + " " + " ...
3
4   }
5   int number1 = ...
6
7   int answer = input.nextInt();
8   int number2 = ...
9   Scanner input = new Scanner(System.in);
```

```
1   import java.util.Scanner;
2   public class AdditionQuiz {
```

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.



Variable i values should be filled according to the instruction on the right within given time

Instruction appears from the bottom related to variables that player has to answer on the left

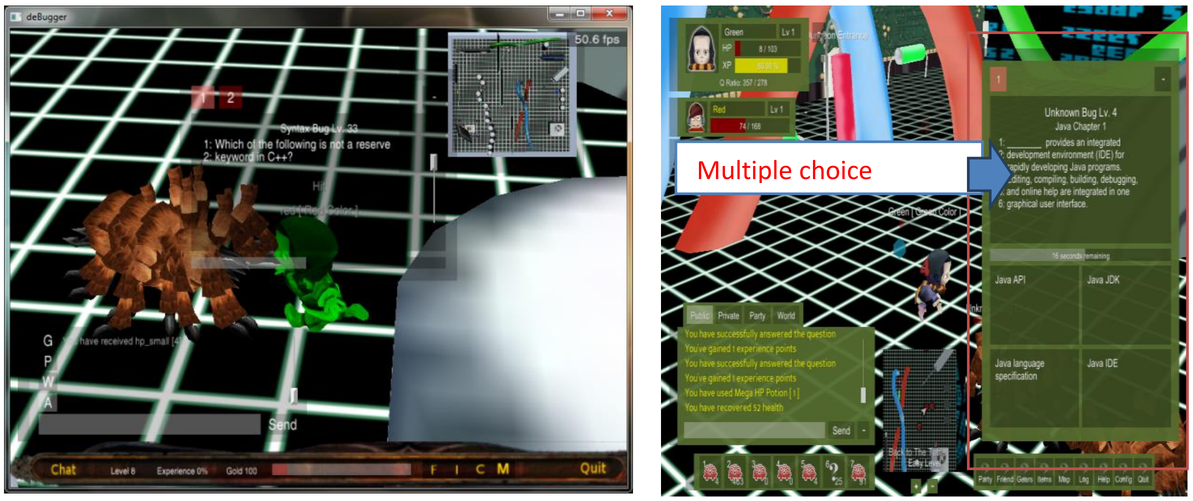
7	45	0	14
8	46	0	14
9	13	0	14
10			

```
7   i++;
8   i = 10;
9   i += 3;
10  i += 2;
```

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.

“Variable” game is designed to teach concepts of variables, data types and operations associated with data types. 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 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).

The DeBugger features a board game and a fighting game to provide multiple choice questions in more exciting way as shown in the Figure 4. Multiple choice style can cover diverse topics from definition, concepts and picking a correct execution result from a given code segments. These questions are organized into levels in the order of variables, data types and operators, selection flow control, repetition flow control, methods, strings, and arrays. So students can play the most suitable level of topics that they have learned. While answering the multiple choice questions, the player engages in a battle with bugs, a board game with other players, a quest for finding big boss bug with other players, or a PvP (Player vs. Player) game with multiple choice questions. Rewards for playing these games are increasing health, getting game gold, increasing level or getting a gift of game items. Game items can be also purchased with game gold and they are designed to add fun to the game and allow players to have desirable options such as greying out two options from the multiple choices, protecting the player's health while being attacked by bugs or upgrading decorative items. See the Figure 5.



Syntax Bug Lv. 33
1: Which of the following is not a reserve
2: keyword in C++?

Unknown Bug Lv. 4
Java Chapter 1
1: ________ provides an integrated
2: development environment (IDE) for
rapidly developing Java programs.
editing, compiling, building, debugging,
and online help are integrated in one
6: graphical user interface.

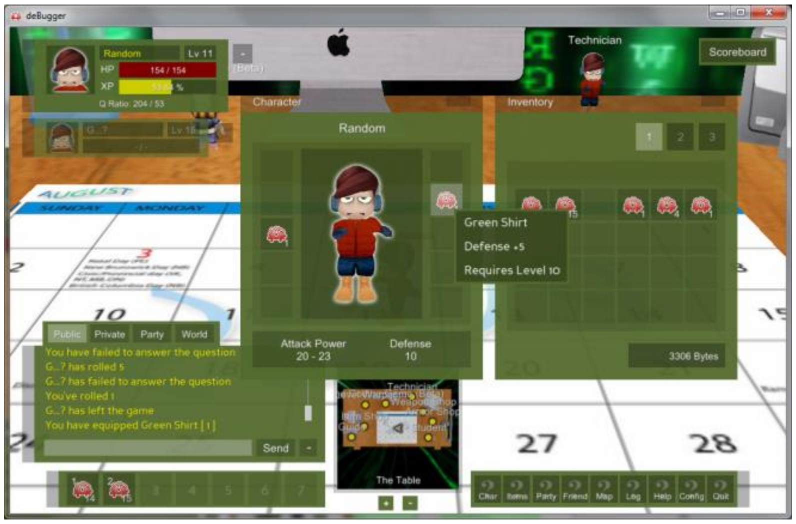
16 seconds remaining

Java API	Java JDK
Java language specification	Java IDE

You have successfully answered the question
You've gained 1 experience points
You have successfully answered the question
You've gained 1 experience points
You have used Mega HP Potion [1]
You have recovered 52 health

Figure 4. A player is fighting with a bug (left). Bug asks a question and the player has to answer (right). Health gets damaged as time lapses. Player's health, level, gold are shown at 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 added easily to allow players to be exposed to further concepts continuously with fun. In addition to the games explained in this paper, there are several more mini games currently available at the DeBugger game[19].



You have failed to answer the question
G...? has rolled 5
G...? has failed to answer the question
You've rolled 1
G...? has left the game
You have equipped Green Shirt [1]

Figure 5. Player can check their inventory for game items. Players can purchase game items using game gold, trade with other players or give a gift to friends.

3.2 Mobile Version

Players can seamlessly continue to play the game using their mobile devices. 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 gameplay (Figure 6).

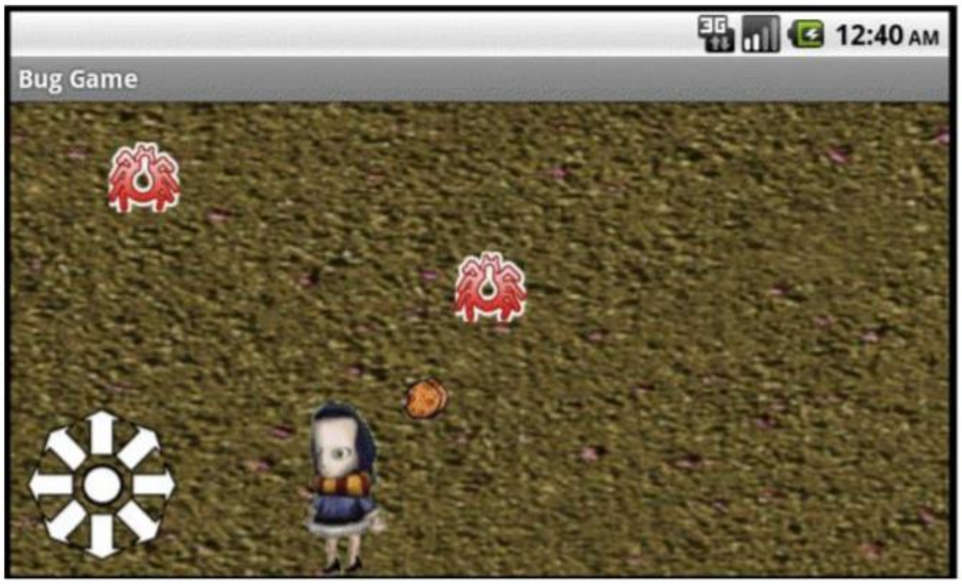
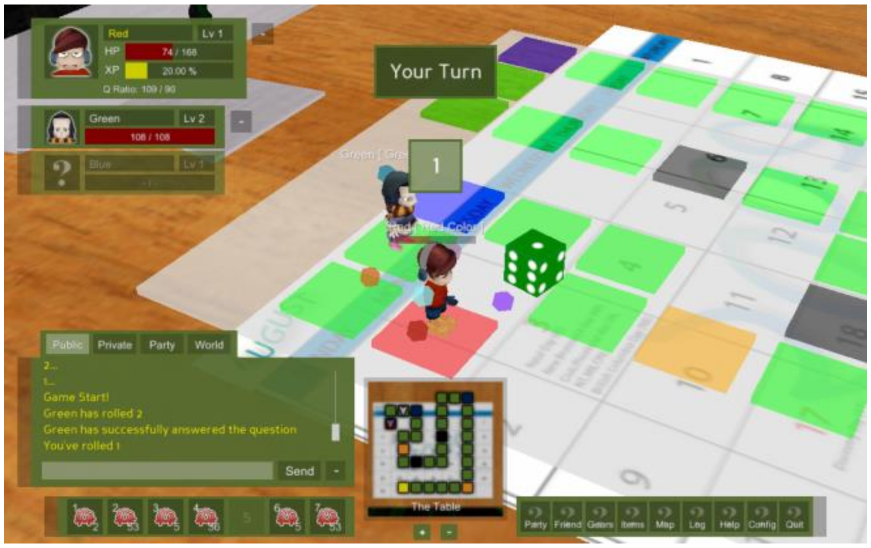


Figure 6. Screen capture of Mobile Debugger. Players can throw stones to the bugs using the touch pad screen,

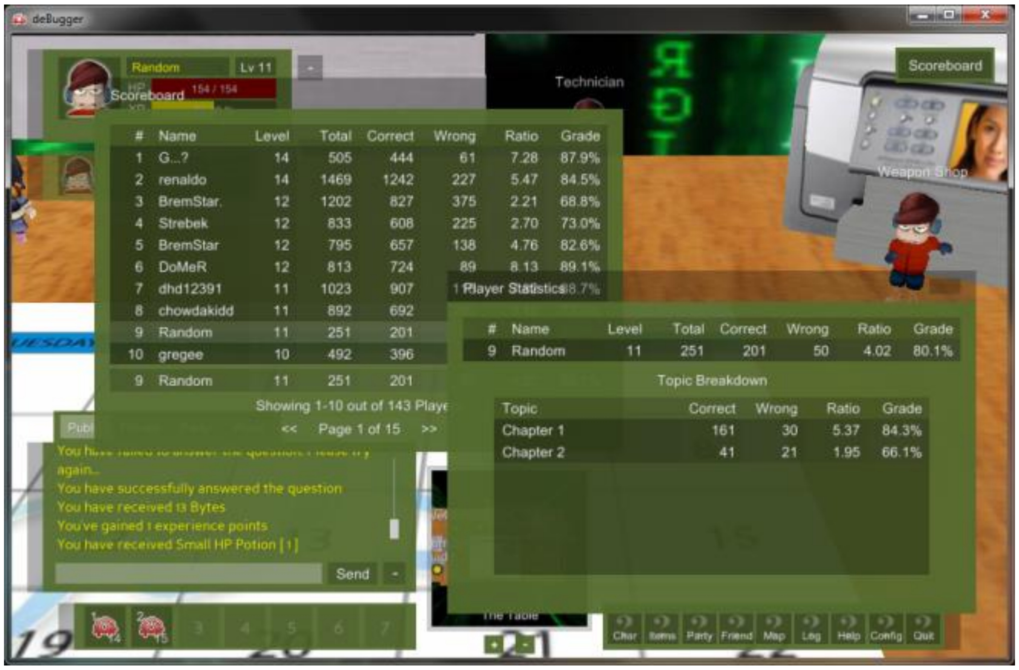
3.3 Social Activity

There are many social activity features within the DeBugger game. Players can play against each other (See Figure 7.) Also they can chat with other players in a few different ways (public chat, private chat, etc). A player can manage a friends list. Player can choose to show their level and other performance to their friends or the public.



2...
1...
Game Start!
Green has rolled 2
Green has successfully answered the question
You've rolled 1

Figure 7. Two players are playing a board game. Each player takes turn, roll the dice, and solve a problem to progress. Winner who arrives the end point faster receives game gold as reward.



#	Name	Level	Total	Correct	Wrong	Ratio	Grade
1	G...?	14	505	444	61	7.28	87.9%
2	renaldo	14	1469	1242	227	5.47	84.5%
3	BremStar.	12	1202	827	375	2.21	68.8%
4	Strebek	12	833	608	225	2.70	73.0%
5	BremStar	12	795	657	138	4.76	82.6%
6	DoMeR	12	813	724	89	8.13	89.1%
7	dhd12391	11	1023	907	1[illegible]	[illegible]	[illegible]8.7%
8	chowdakidd	11	892	692	[illegible]	[illegible]	[illegible]
9	Random	11	251	201	[illegible]	[illegible]	[illegible]
10	gregee	10	492	396	[illegible]	[illegible]	[illegible]
9	Random	11	251	201	[illegible]	[illegible]	[illegible]

Showing 1-10 out of 143 Playe

<< Page 1 of 15 >>

#	Name	Level	Total	Correct	Wrong	Ratio	Grade
9	Random	11	251	201	50	4.02	80.1%

Topic Breakdown

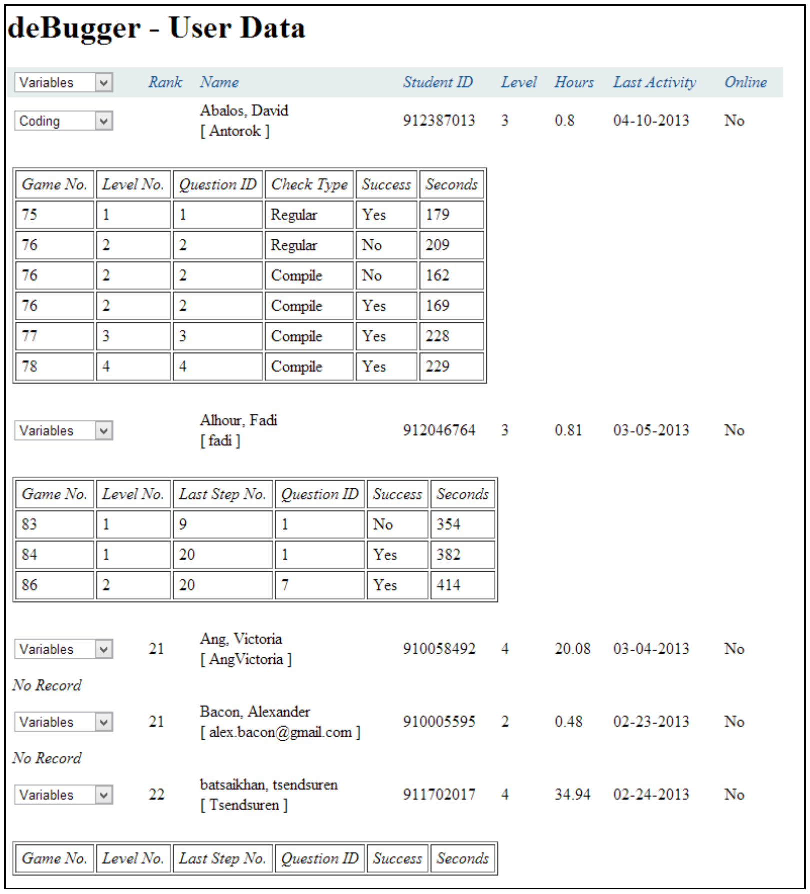
Topic	Correct	Wrong	Ratio	Grade
Chapter 1	161	30	5.37	84.3%
Chapter 2	41	21	1.95	66.1%

You h[illegible]
again...
You have successfully answered the question
You have received 13 Bytes
You've gained 1 experience points
You have received Small HP Potion [1]

Figure 8. Player's performance is measured in diverse ways. And players can see other players' performance unless blocked by the player.

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 as shown in the Figure 8. Currently, we display players' game IDs with scores but we consider other ways to display scores to encourage players in different levels or promote competitions. For example, in order to promote friendly competition among classes/sections or schools, we consider to display players' section or school information on the board.

Social gaming and MMORPG are relatively recent game genres as they gain popularity over the availability of Internet. Social interaction within the game generates excitement of the game which allows popular games to easily develop communities of multimillion players. Once a community of learners is created, students can learn not only from the 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.



deBugger - User Data

Variables	Rank	Name	Student ID	Level	Hours	Last Activity	Online
Coding		Abalos, David [Antorok]	912387013	3	0.8	04-10-2013	No

Game No.	Level No.	Question ID	Check Type	Success	Seconds
75	1	1	Regular	Yes	179
76	2	2	Regular	No	209
76	2	2	Compile	No	162
76	2	2	Compile	Yes	169
77	3	3	Compile	Yes	228
78	4	4	Compile	Yes	229

Variables		Alhour, Fadi [fadi]	912046764	3	0.81	03-05-2013	No

Game No.	Level No.	Last Step No.	Question ID	Success	Seconds
83	1	9	1	No	354
84	1	20	1	Yes	382
86	2	20	7	Yes	414

Variables	21	Ang, Victoria [AngVictoria]	910058492	4	20.08	03-04-2013	No

No Record

Variables	21	Bacon, Alexander [alex.bacon@gmail.com]	910005595	2	0.48	02-23-2013	No

No Record

Variables	22	batsaikhan, tsendsuren [Tsendsuren]	911702017	4	34.94	02-24-2013	No

Game No.	Level No.	Last Step No.	Question ID	Success	Seconds

Figure 9. A screen shot of the web site where teachers can see all sorts of students' performance in the DeBugger

3.4 Player Data

The DeBugger is not only for students. The data collection tool assesses students' learning outcomes automatically and provides useful information (e.g. grading of homework) back to teacher (See Figure 9.) This is very useful and efficient in that teachers save time for grading and students receive feedback immediately. Teachers can see all of the student's performance in diverse analytical views (number of hours, levels, correction ratio, etc).

3.5 Game Architecture

Since the DeBugger game is an online game, we have a game server that runs all the time and supports the virtual DeBugger world persistently. The game server is connected by all the game 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 persistent. 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 10 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.

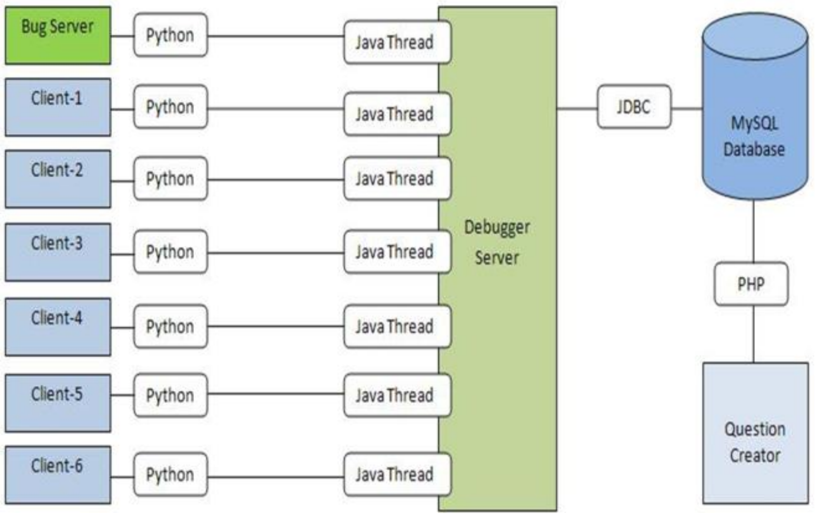


Figure 10. DeBugger Architect. It shows the relationship between a Client and the Server

Stable support of the DeBugger game server is critically important for MMORPG game. And we evaluated the scalability utilizing Clouds Computing acting as individual clients and DeBugger server stably support 100 concurrent players more than 100 heartbeat per second performance.

4. Project Efficacy Testing and Results

4.1 Testing Setup

The DeBugger game has been used at SFSU CSc 210 Introduction to Programming using JAVA during the Spring 2012 (pilot test) & Spring 2013 (official efficacy testing) semesters. During the Spring 2013, 29 student participants were recruited from 4 introductory Computer Science programming classes. Seventeen students were introduced to the DeBugger for the first 2 months as a part of their learning (Experimental Group) while 12 students were not given such exposure to the DeBugger (Control Group). All students were required to study textbook chapters. The experimental group students were asked to use the DeBugger game and collect 60 correct answers while the control group students were asked to submit a summary of each chapter in a 3 page essay format. We estimated that it would take at the minimum 1 hour a week to complete any one of tasks.

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 access to the DeBugger for the next 8 weeks. To examine the effectiveness of playing the 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.

4.2 Results

Our preliminary data analyses have shown positive impact of playing the DeBugger on student learning outcomes. Figure 11 illustrates participants' mean percent correct on the pre- and post-tests.

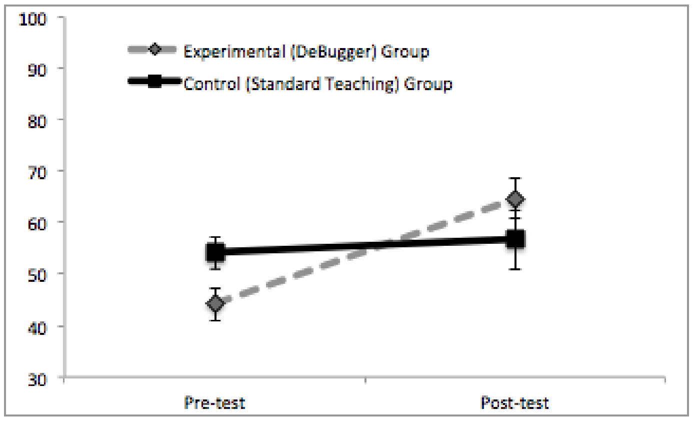


Figure 11. Students' Mean Percentage Scores on Pre- and Post-tests.

To analyze evaluation results, ANOVA (ANalysis Of VAriance) statistical hypothesis testing was used. All symbols such as *F* (F-test statistic ratio), *p*, η^2 , *M*(mean), *SD*(standard deviation), and *t* are standard statistical symbols used in ANOVA. 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 the 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: $M = 56.68$; $SD = 20.6$), $t(11) = 0.33$.

Usability and learning impact of the DeBugger is continuously being evaluated. Currently, the DeBugger game is being tested at SFSU CSc 210 Introduction to Programming using JAVA during this Spring 2014. There are total of 100 student participants from 2 sections (58 + 42). 88 students participate in using the DeBugger (Experimental Group) while 12 students don't (Control Group). Since it is still in the middle of the semester, full analysis data is not available yet. However, we have noticed one significant data showing that an average number of problems students in the experimental group solved using the DeBugger is far above the required number 60. Out of 88 participants, 87.5% students collected 110% or more correct answers. Only 11 students collected 110% or less correct answers. The average of 88 students collected 295% correct answers. In order words, so far 120 correct answers were required for students to collect for chapter 1 and 2, but each student voluntarily collected about 360 correct answers on average. This demonstrates that the DeBugger as a MMORPG can help increase students' study hours painlessly and eventually improve their learning outcome.

Besides getting positive data analysis results, we received many reassuring comments from participants. Following 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. "

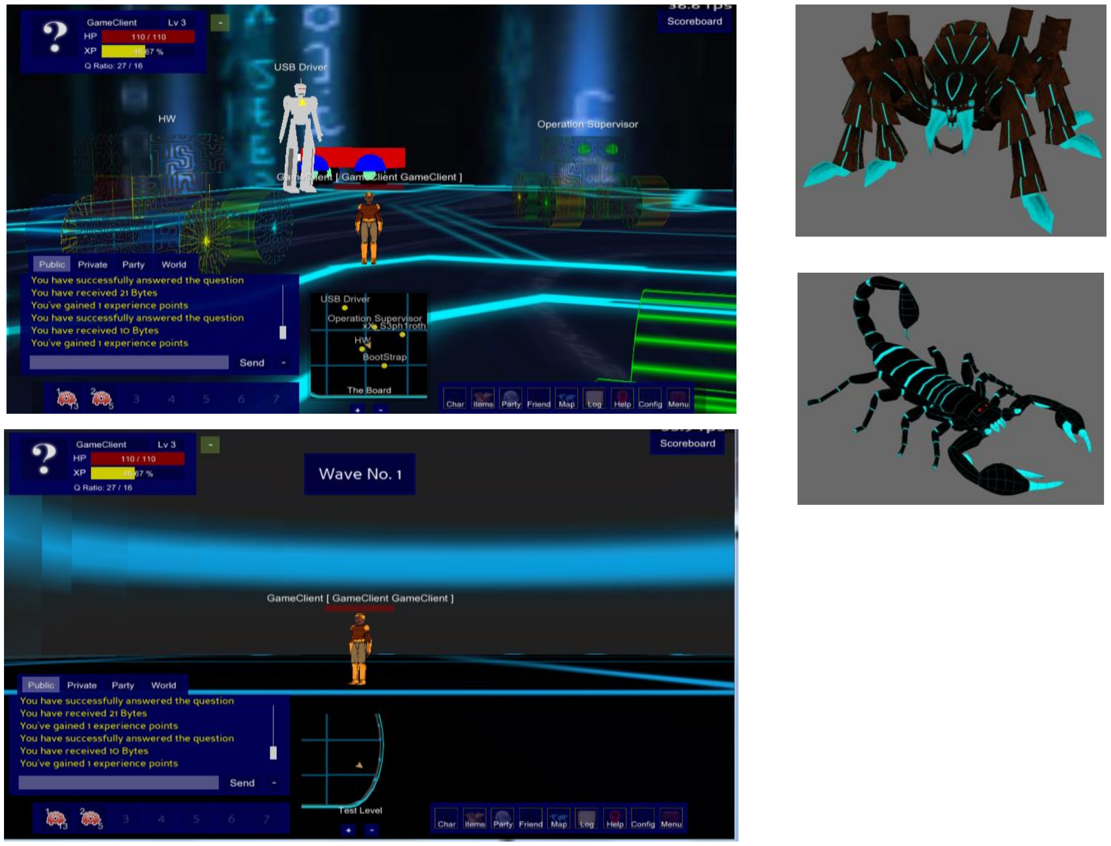
"There were two other players from CSc 210 that played Debugger today. We played the bugs and board game for about 30 - 40 minutes. It was really fun, and we talked a lot about programming and school and stuff. When we both played the board game at level three, and it's more fun at a harder level because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

5. On-going Development

Further developments are in progress. First addition is for practicing selections (if-else and switch branches) and flow control concepts (e.g. loops). Currently, we are developing a game called "*DeBugger - Infinite Loop*", an expansion to the DeBugger. The story of Infinite Loop takes place inside a computer that can be reached from the main DeBugger hub area, the Table. Infinite Loop will have its own Hub, the motherboard, where on the first arrival, the player is told about the crisis, that the computer has gone into an infinite loop due to a bug, and is asked for help. The gameplay of Infinite Loop will mainly be delivered by an improved version of the existing battle system currently in place, as well as by a set of mini-games demonstrating the objective programming concepts. Figure 12 and 13 show new features of Infinite Loop.

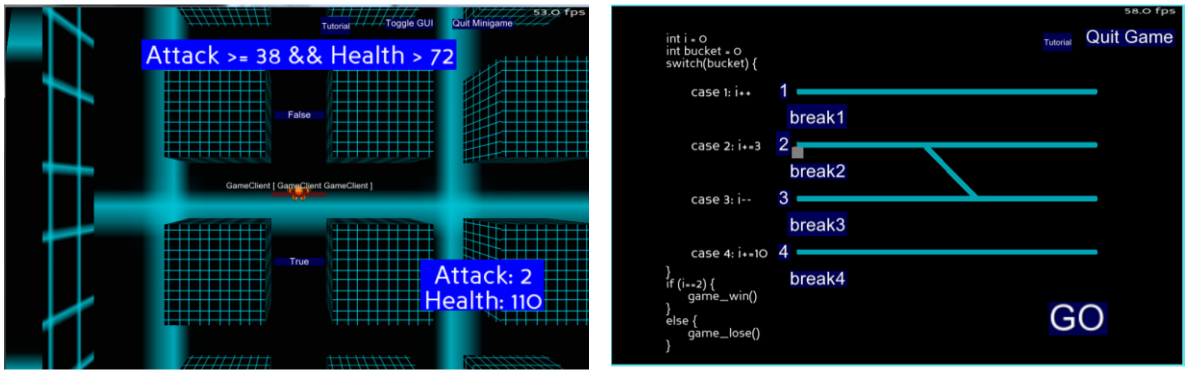


You have successfully answered the question
You have received 21 Bytes
You've gained 1 experience points
You have successfully answered the question
You have received 10 Bytes
You've gained 1 experience points

Figure 12. DeBugger - Infinite Loop. Hub area (top-left). New coliseum battle system (bottom-left). Two bug models in the Tron theme (right).

The story of Infinite Loop is composed of 3 main sections, taking place in 3 different areas, a training facility/coliseum mainly focusing on if/else/switch/case, a light-train on the way to the bug mainly focusing on loops, and a final area where the player party will have to face the bug causing the infinite loop. The player will begin in the original game's hub area and talk to an NPC to move to the new hub area -- the motherboard. From the motherboard, they will go to the USB driver to enter into the coliseum. They will do a mixture of coliseum battles and the if-else maze mini-games, eventually moving onto the light train area. From there, they will complete the fight quests and light train mini-games. Then they will move to the final dungeon. They will complete the dungeon and then finish the game.

Infinite Loop is designed to add to the existing DeBugger not only mingames but also new game features such as generating new question types for if-else and loops, caracter customization, and GUI manager. As for minigames, coliseum battle system, If-Else maze, and Swich minigames are near completion. In the If-Else minigame, the player starts at a random intersection in the maze. The correct path is randomly generated and answering a question moves the player up, down, left, or right. The goal is to answer 5-10 if-else questions in a row correctly to find the exit. In the Switch minigame, a switch statement question is presented. In it, there are case statements that modify a value 'i'. The goal is to guide a bucket through the rails so that 'i' would be equal to a value that causes the boolean statement to be true. The player should do this for 5 questions to exit. Answering one question correctly allows the player to continue to the next question. Figure 13 shows screen shots of two minigames. With additional features, the DeBugger will help students more and better to understand a wide range of JAVA language concepts.



```
int i = 0
int bucket = 0
switch(bucket) {
    case 1: i++
    case 2: i+=3
    case 3: i--
    case 4: i+=10
}
if (i==2) {
    game_win()
}
else {
    game_lose()
}
```

Figure 13. DeBugger - Infinite Loop. Top-top view of If-Else Maze game (left). Switch minigame (right).

6. Conclusion

We introduced a MMORPG game, DeBugger, which 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 results have shown that the impact of the game is effective and socially cooperative as a virtual lab and space for peer instruction.

Acknowledgement

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.

Bibliography

1. 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.
2. 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.
3. Jens Bennedsen and Michael E. Caspersen, Failure rates in introductory programming, SIGCSE 2007, pp 32-36.
4. Computing Education for 21st Century (CE21) Program Solicitation NSF 12-527 http://www.nsf.gov/pubs/2012/nsf12527/nsf12527.htm
5. 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.
6. Lee, Irene, et al., Computational Thinking for Youth in Practice, ACM Inroads, Vol. 2, No. 1, March 2011.
7. 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.
8. Ng, Brian D. and Peter Wiemer-Hastings, Addiction to the Internet and Online Gaming, Cyber Psychology & Behavior 8.2 (2005)
9. 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
10. Yee, N. (2002) Understanding MMORPG Addiction.
11. 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.
12. 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.
13. 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.
14. 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.
15. Christudason, A. Successful Peer Learning, http://www.cdtl.nus.edu.sg/success/sl37.htm
16. Falchikov, N., Blythman, M., Learning together: peer tutoring in higher education.
17. Johnson, D.W.; Johnson, R.T.; & Holubec, E.J. (1993). Circles of Learning. Edina, MI: Interaction Book Company.
18. Kaufman, D.B.; Felder, R.M.; & Fuller, H. 'Peer Ratings in Cooperative Learning Teams'. Proceedings of the 1999 Annual ASEE Meeting, ASEE, Session 1430'.
19. 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