Why Not Have Fun While Learning: Using Programming Games in Software Programming Education

Ju Long
jl38@txstate.edu
Department of CIS & QM
Texas State University – San Marcos
San Marcos, TX 78666

Abstract

In this research, we examine how computer-based programming games and contests could be used in the information technology education to increase the learning effectiveness for learners. We conduct a field study in a unique community of software developers who participated in IBM Robocode game. In our field study, we address the following research questions: (1) how do computer-based games influence the learner's learning outcomes? Did the learner's programming skills and knowledge improve after participating in the programming game? (2) Is the programming game appealing to learner groups with different ages, education backgrounds and skill levels? (3) What do the learners like most about the programming game? (4) What are the factors that influence the learner's motivation to engage in the programming game? (5) What programming stages are of more intrinsic fun value and how to design the programming game accordingly? Through our case study, we found that (1) computer-based programming games and contests could significantly increase the intrinsic motivation of the learners across all learning levels, ages and education backgrounds. (2) These programming games could improve the learning effectiveness very efficiently. (3) Different stages of programming have various intrinsic fun values. Researchers and practitioners could design programming games and contests accordingly to improve the intrinsic fun factors.

Key Words: Information Technology Education, Software Programming Education and Training, Programming Games, Computer Games, Intrinsic Motivation

1. Introduction

Improving learning effectiveness has been a constant challenge in the software programming education. A learner's performance is affected by both her abilities and motivations, and one of the primary tasks educators have is to motivate learners to perform to the best of their abilities. One of the instructional methods that have been used to keep learners engaged in learning is the game-based exercise. In this research, we examine how computer-based programming games could be used in the software programming education to increase the learning effectiveness for learners.

To examine how computer-based games could improve learner's intrinsic motivations and learning experiences, we

conducted a field study in a unique community of software developers who participated in the IBM Robocode game. IBM Robocode teaches developers the Java programming language in a game-like format. In our field study, we address the following research questions: (1) how do computer-based games influence learner's learning outcomes? Did the learner's programming skills and knowledge improve after participating in the programming game? (2) Is the programming game appealing to learner groups with different ages, education backgrounds and skill levels? (3) What do the learners like most about the programming game? In another words, why is the programming game fun? (4) What are the factors that influence the learner's motivation to engage in the programming game? (5) What programming stages are of more intrinsic fun value and how to design the programming game accordingly?

Through our case study, we found that (1) computer based programming games and contests could significantly increase the intrinsic motivation of the learners across all learning levels, ages and education backgrounds. (2) These programming games could improve the learning effectiveness very efficiently. (3) Different stages of programming have various intrinsic fun values. Researchers and practitioners could design programming games and contests accordingly to improve the intrinsic fun factors.

For academic researchers, our research could lead to a better understanding of how to improve the learning effectiveness in the software programming education using computer based games and contests. For instructors and practitioners in the information technology education field, our research, especially the real world field study, could provide practical strategies and best practice examples on how to integrate programming games into the IT education and training.

2. Theory Background

Using Computer Games In General Education

Computer games are seen as a means of encouraging learners who may lack the interest to learn (Klawe 1994). Games are also a means to enhance the self-esteem for the learners who may lack the confidence in learning (Ritchie and Dodge 1992; Dempsey et al. 1994). When games are used in training and educational settings, it is suggested that they can reduce the training time, provide more opportunities for practice, and enhance knowledge acquisition consequently (Brownfield and Vik 1983; Ricci 1994). Computer games also led to positive results in long-term learner retention by improving learning interests (Randel et al. 1992) and more focused attention, because the students enjoyed the approach (Ricci 1994).

Computer games are said to be particularly effective when 'designed to address a specific problem or to teach a certain skill' (Griffiths 2002). Games have been used to encourage learning in curriculum areas such as math, physics and language arts, where specific objectives can be stated (Randel et al. 1992). Games could be applied in this area as well.

Computer games have also been used as a means to foster the learners' understanding of theoretical models and interaction effects and to support the development of team, social, communication and resource sharing skills (Ritchie and Dodge 1992; Berson 1996; Helliar et al.2000; Hollins 2003; Squire et al. 2003). Since building team skills and communication skills are important components in the information technology education, incorporating computer games could be beneficial as well.

Researchers have also studied why computer games could promote learner's learning motivation and the final learning effectiveness. Computer games are typically fast and responsive. They could provide a rich variety of graphic representations to generate a wide range of options and scenarios not possible with non-computer games (Pensky, 2001). For instance, simulation games are flexible and complex enough to cater for different learning styles (Sedighian 1994; Kirriemuir 2002). In addition, computer games can

deal with infinite amounts of contents and afford different levels of challenge. The instant feedback and risk-free environment of computer games invite exploration and experimentation and stimulate curiosity, discovery learning and perseverance (Kirriemuir 2002). Computer games also encourage visualization, experimentation and creativity in finding new ways to tackle the game (Betz 1995; Gee 2003).

Using Computer Games in End User Computing Training

Research in the information technology field has also explored how computer games could improve the technology training effectiveness in the end use computing environment. In a study on the ease-of-use perceptions on new technologies, Venkatesh (1999) suggests that a game-based program will amplify the ease-of-use beliefs. They presented empirical results comparing a "traditional" training environment with a game-based training environment, where the latter was constructed so as to be more enjoyable to the users. As posited, ease-of-use perceptions were higher with the game-based training group than with the traditional training group. For another example, in a computer training setting, Martocchio and Webster (1992) suggested that intrinsically motivated individuals approach technology with a more imaginative style that encourages skill development, and learn new skills more effectively. Among other research, Sansone et al. (1989) also found that game-based training format led to higher levels of enjoyment and interest in further information.

However, the studies we reviewed above have not systematically studied the game-based program in the software development training and education setting, which is the central theme of our research.

3. Research Methods and Data Collection

The field research site we chose is the IBM Robocode game community. The Robocode project is created by Mat Nelson to promote the learning of Java as a programming language. Participants in Robocode community are all real world learners interested in the Java programming, albeit with various levels of expertise and experiences. To play Robocode game, each developer creates a "robot" program using the Java programming language. The Robocode framework defines the basic physical rules every robot has to follow and provides a re-usable object structure to ease the development. Developers' robots then compete in Internet-based "leagues" where each robot tries to search and destroy other robots while protecting itself. The winning robots are the ones that have utilized the best strategies and have the most optimized implementations. We selected Robocode as our research site because of its education value to the participants. Robocode is a game that requires participants to put in a significant amount of efforts to constantly maintain and develop their "Robots". During the process, participants learn Java language at every stage of software development process, including algorithm design, architecture, implementation, optimization, testing and bug fixing.

Robocode has been highly successful since IBM made it publicly available in July 2001. Within 8 months, the program has been downloaded more than 120,000 times. This popular programming game provides us three distinctive benefits to study the programming game's effectiveness in the information technology education.

First, unlike simulations conducted in classroom settings, Robocode is a real-world programming game with real world participants, which could improve the validity of our research. Second, Robocode has a large participant base, which could give us a large population to draw our sample from. The participants in our sample could have various programming skills and education levels, thus we could increase the validity of our research. Third, in Robocode, all participants work on the same software and the same platform using the same programming language. This homogenous developing environment means that the development tasks and processes are virtually the same to each developer. We can then focus on studying

the programming games and the learning outcomes without the intervention caused by different programming languages or programming environments.

We use the survey on Robocode participants as our main data collection method. Robocode community has maintained a very active online discussion forum for Robocode participants. Our survey sample is randomly drawn from the forum. We've sent out a total of 500 surveys to the developers and excluding two invalid responses, we generated 83 valid responses (A response rate of 17%)

4. Data Analyses and Results

Programming games could effectively improve learning outcomes

We found that Robocode game is a very effective tool to promote the self-motivated learning in the information technology education, especially in the software programming education and training. Such self-motivated learning is considered by many as the best way to learn (Lepper & Malone, 1987). In our field study, we specifically test how effective Robocode game could help learners to learn new programming knowledge and skills. About 80% of the participants report that their programming skills have increased through participating in the Robocode. Among these participants, more than 20% said that their skills have improved significantly and about 60% report that their skills have increased to some extent. Only 20% report that their skills stayed about the same. (See table 1). This result strongly suggests that programming games are effective in promoting self-motivated learning.

Because our study is designed as an explanatory case study instead of a controlled experiment, our results are exclusively based on the participant survey. In our future research, besides the participants' self-reported learning effectiveness, we plan to include subjects in the Robocode game and test their learning outcomes using the controlled field experiment method.

Programming games can be used across all age, education and expertise level

Participants in our sample come from all age groups (See Table 2). The majority of them (75%) are between the age of 18 and 34, with about 40% of the participants in the 25 to 34 age group, and another 20% of the participants in the 35 to 49 age group. Our results strongly indicate that programming games are attractive both to young learners and also to older and often more experienced learners. We also did a correlation test between the developer's motivation to participate in the game and their age groups and did not find any significant correlation (see table 3). This result implies that programming games could be applied in the education and training to learners across all age groups. Notably, our finding are consistent with the results of a study by Venkatesh (1999), who examined game-based versus traditional training, and did not find any evidence of a moderating effect of age on the effects of the training method.

The education levels of the developers/learners in our sample are quite high. In our sample, nearly 48.2% of the participants have some graduate school education or completed graduate school. Another 42.2% have some college education or completed college school. This finding implied that programming games could not only be used in college level technology education and training programs, but also be applied in graduate level education and corporate training programs as well. Again, we conducted a correlation test on the education level and the motivation to participate in the game. We did not find any significant correlation between these two variables (see table 3). It further confirms our finding that programming games could be appealing to participants in various education levels.

We also conducted a correlation test on the participant's motivation to engage in the game with their expertise levels. To identify the participants' level of expertise, we take advantage of Robocode's existing system to differentiate the participants' skills. Upon participating in Robocode, learners would be asked to assign

themselves into three play leagues based on their programming skills. The beginners are participants with no Java programming experiences before. The intermediate participants are those with some experiences (less than six months) in Java programming or those who are very experienced with another programming language but not Java; the advanced participants are experienced and skilled Java programmers. Using the same criteria, we asked the participants to rate their skill levels in the survey. Our results show that our sample includes developers from all three levels: approximately 23% of the developers are beginners, approximately 40% are intermediate developers, and about 37% are advanced developers. Our results show that the stratified skill levels listed above do not significantly influence the motivation to participate in the game (See table 3).

The above analyses confirm our proposition that learners' motivation to participate the game-based programming training and education is not influenced by their expertise, age and educational levels. (See Table 3)

Motivation factors that influence the learner's effort in programming games

One of our main research questions is to study the factors that keep the learners engaged in the software programming game. A clear understanding of these factors could help us determine strategies to keep participants engaged in the programming games and improve the effectiveness of this method. We include a wide range of motivational factors, including simply for the fun of the game, to be a winner of the game, to compete for the prize in the contest, to learn new programming skills and to gain recognition among peers (see Table 4).

Our results show that the intrinsic fun of the game is the most important motivation factor for the learner to engage in the game. 87.5% of the participants in our sample chose fun of programming games as one of their participation motivations. Another important motivation factor is the fun to learn new programming skills. About 54% of the participants chose that as an important reason to participate in the programming game. Compared with the above two intrinsic motivators, extrinsic motivators, such as "to win the game", "to win the prize in the contest" and "to gain peer recognition", are less significant. Only about 32.5% of the participants chose the "to win the competition" as one of the motivations to participate in the programming game. Even less participants (11.3%) chose "to win the prize" as a motivation. "To gain recognition among the peers" is not an essential motivation either. About 16.3% of the participants chose it as a reason to participate in the programming game.

Why the programming game is engaging?

We examine further the reasons why the programming game could be fun for the participants (See table 5). Among the reasons we list, "To be able to solve problems on my own" is chosen as the most important factor to contribute to the enjoyment of programming. 65.4% of the participants chose it as very important, and 22.9% of the participants chose it as important. "To be able to be creative" is chosen as the second important factor to influence intrinsic motivations. More than 74% of the respondents chose the creativity as an important factor - with 53.1% of the participants chose it as a very important factor, while 21% chose it as an important factor. The above two reasons could both be categorized as indicators of autonomy. Our results conform to the previous theories on motivation that the more autonomy the learner has, the more fun the learning process will be to the learners, and the more motivated the learners will be (Deci, 1975).

"To be able to put skills in use" has also been selected as one of the most important reasons why the programming game is fun to the participant. More than 65.3% of the respondents chose "put skills in use" as either very important (28.8%) or important (37.5%). This reason is an indicator of competence. Another indicator is "to be able to learn new skills". About 55.7% of the respondents chose "to be able to learn new skills" as a very

important (22.8%) or an important (32.9%) factor in determining their intrinsic motivations. It implies that to make the learners feel competent is critical for the programming game to be attractive to the learners.

Various programming stages and their intrinsic fun levels in the game

We examined how various stages in the software development process have different levels of fun. We then could design the programming games accordingly to maximize the fun level of the programming games. We divide the programming process into the following stages, including discover the algorithms, design the architecture, write the code, and test and debug (see Table 6).

Among all the stages, discovering algorithms is chosen as the most enjoyable stage: more than 80% of our participants enjoy this task (52.5% enjoy it very much, and 27.5% enjoy it to some extent). As we have discussed in the previous section, to be able to be creative is an important reason for the programming game to be fun. Discovering algorithms is a very creative process. Learners could employ a wide range of knowledge and expertise, and try out various solutions. Usually developers can design the algorithms based on their own decisions. Thus, it provides a high level of enjoyment.

Similar to the stage of algorithm design, the stage of designing the architecture has strong intrinsic fun as well. About 71% of the respondents enjoy it. Among them, about 34.2% think it is very enjoyable, and 36.7% think it enjoyable. Designing the architecture is regarded as a very prestigious task and could be a strong signal of developers' high level of expertise. Thus it could induce a sense of competence. Designing the architecture also let the developers have complete control on how to design the project, thus a high level of autonomy is ensured. Because of both high levels of autonomy and competence, this stage is of high intrinsic fun value as well.

Compared with the more creative and fun stages of discovering algorithm and designing the architecture, writing the code and testing and debugging the code have much less intrinsic fun value. Only 44.3% of the respondents enjoy writing code; and only about 21.7% of the respondents enjoy testing and debugging, while 78.2% do not like testing or debugging or are neutral about it. The main reason could be the relatively low level of autonomy and competence in those development stages. Creativity, a critical factor in determining the fun level of tasks, is relatively low in coding and debugging. Developers usually have to follow certain syntax rules in coding and debugging. The expertise involved in these two stages of development is often specific, leaving little space for the developers to be creative. As researchers, we could design the programming games according to the different fun level of the programming stages, which we will discuss in detail in the following section.

5. Research Implications

A better understanding of how to incorporate programming games in the information technology education would be valuable not only to practitioners responsible for improving the learning effectiveness in the software education and training, but also to researchers examining the methods through which the information technology education could be enhanced.
Facilitate information technology education through programming games

As the number and complexity of new technologies available to learners increase at a fast speed each day, the importance of self-motivated learning also increases. Firms need their future developers to continually adjust to new technology advances and adapt it to their day-to-day work. By leveraging game-based training methods, we could make IT education and training courses more enjoyable while continuing to encourage students' knowledge acquisition. As we have shown, programming games are very effective in increasing learners' efforts in the learning process. We have proved that the more the learners enjoy programming, the more effort they will put into learning, shown as the more time they spend and more sophisticated programs they develop.

Instructors hoping to improve learners' learning outcomes could leverage the programming game as a fun training tool and enhance the learners' learning experiences and outcomes.

Based on our findings, we could learn some implications when designing programming games. First, educators and instructors could pay more attention to the less enjoyable tasks and make them more fun by incorporating programming games into the educations and training. The rationale behind this strategy is to recognize the different intrinsic fun levels of development stages and utilize programming games to make the less enjoyable tasks more fun. As we have shown in our results, development stages such as algorithm design and being an implementation architect have very high intrinsic motivational strength for developers. Learners would work hard in those stages simply for the enjoyment and fun during the process. However, for the stages with low fun value, such as debugging and testing, we could incorporate programming games and improve the learner's learning motivation.

Second, it is important that games are used to facilitate tasks appropriate to learners' level of maturity in the skill (Din and Calao 2001). The start-up of the games should be kept simple, since the learners' thresholds of interest and concentration may be low. The instructions of the games should also be kept simple to minimize levels of frustration and time spent learning the rules of the game. The designers/educators could divide the task into shorter modules so that learners could have more instant gratification and increase the sense of autonomy and competence. The designers could vary between short modules (to maximize the likelihood of satisfactory outcomes) but also make longer sessions available (to encourage involvement). Last, but not least, the game should be able to provide different levels of challenges and cater to different learning levels. Robocode's league system could be an excellent example on the implementation.

6. Conclusion and future research

This research examines how programming games could be used to enhance the software programming education and training. Based on our field study of a real world programming game – Robocode, we demonstrated that programming games could improve the learner's motivation in the learning process and generate better learning effectiveness. We also provide several evidences that programming games can be applied to learners with various skills, experiences and ages. Furthermore, we examined the factors why programming games are appealing to the learners. We also studied the different fun levels in various programming stages and suggested game design strategies according to our findings. Our research is of value to the researchers in the IT education and training as well as the practitioners in the field.

Our research is an exploratory field study. In the future research agenda, we will conduct a field experiment in the Robocode game and study how programming games could improve the learning effectiveness in a controlled lab format. In addition, it is interesting to note that all of the participants in our current study are male. This may be partly because that Robocode is not a game that is appealing to women learners. It also reflects the enormous gender gap in the current IT professional population. Results from our sample could only represent the experiences and characteristics of male learners. Therefore, another future topic will be how to create programming games that are appealing to women learners.

7. Reference

Berson MJ (1996). Effectiveness of computer technology in social studies: a review of the literature. Journal of Research on Computing in Education, 28(4), 486–499.

Betz JA (1995). Computer games: increase learning in an interactive multidisciplinary environment. Journal of Educational Technology Systems, 24(2), 195–205.

Brownfield S, Vik G (1983). Teaching basic skills with computer games. Training and Developmental Journal, 37(2), 52–56.

Deci, E. L. Intrinsic Motivation. New York: Plenum Press, 1975.

Dempsey JV, Rasmussen K, Lucassen B (1994). Instructional gaming: implications for instructional technology. Paper presented at the Annual Meeting of the Association for Educational Communications and Technology, 16–20 February 1994, Nashville, TN.

Din FS, Calao J (2001). The effects of playing educational video games on kindergarten achievement. Child Study Journal, 31(1), 95–102.

Gee JP (2003). What video games have to teach us about learning and literacy. New York: Palgrave Macmillan.

Griffiths MD (2002). The educational benefits of videogames. Education and Health, 20(3), 47–51.

Helliar CV, Michaelson R, Power DM, Sinclair CD (2000). Using a portfolio management game (Finesse) to teach finance. Accounting Education, 9(1), 37–51.

Hollins P (2003). Playing is the new learning. E.Learning Age, December–January, 16–19.

Kirriemuir J (2002). The relevance of video games and gaming consoles to the higher and further education learning experience. April 2002. Techwatch Report TSW 02.01. At www.jisc.ac.uk/index.cfm?name=techwatch_report_0201

Klawe MM (1994). The educational potential of electronic games and the E-GEMS Project. In T Ottman and I Tomek (eds) Proceedings of the ED-MEDIA 94 World Conference on Educational Multimedia and Hypermedia. Panel discussion 'Can electronic games make a positive contribution to the learning of mathematics and science in the intermediate classroom?' AACE (Association for the Advancement of Computing in Education), Vancouver, Canada, 25–30 June 1994.

Lepper, M.R., and Malone, T.W. Intrinsic motivation and instructional effectiveness in computer-based education. In Snow, R.E., and Farr, M.J. (Eds.), Aptitude, Learning and Instruction, Hillsdale, NJ: Erlbaum, 1987, 255-286.

Martocchio, J.J., and Webster, J. Effects of feedback and cognitive playfulness on performance in microcomputer software training. Personnel Psychology, 45 (1992), 553-578.

Prensky M (2001). Digital game-based learning. New York: McGraw-Hill.

Randel JM, Morris BA, Wetzel CD, Whitehill BV (1992). The effectiveness of games for educational purposes: a review of recent research. Simulation and Gaming, 23(3), 261–276.

Ricci KE (1994). The use of computer-based videogames in knowledge acquisition and retention. Journal of Interactive Instruction Development, 7(1), 17–22.

Ritchie D, Dodge B (1992). Integrating technology usage across the curriculum. Paper presented to the Annual Conference on Technology and Teacher Education, 12–15 March 1992, Houston, TX.

Sansone, C.; Sachau, D. A.; and Weir, C. Effects of instruction on intrinsic interest: the importance of context. Journal of Personality and Social Psychology, 57, 5 (1989), 819-829.

Sedighian K (1994). Playing styles for computer and video games. In T Ottman and I Tomek (eds) Proceedings of the ED-MEDIA 94 World Conference on Educational Multimedia and Hypermedia. Panel discussion 'Can electronic games make a positive contribution to the learning of mathematics and science in the intermediate classroom?' AACE (Association for the Advancement of Computing in Education), Vancouver, Canada, 25–30 June 1994.

Squire K, Jenkins H, Holland W, Miller H, O'Driscoll A, Tan KP, Todd K (2003). Design principles of next-generation digital gaming for education. Educational Technology, September–October, 17–23.

Venkatesh, V. Creation of favorable user perceptions: exploring the role of intrinsic motivation. MIS Quarterly, 23, 2 (June 1999), 239-261

Venkatesh, V., and Speier, C. Computer technology training in the workplace: a longitudinal investigation of the effect of mood. Organizational Behavior and Human Decision Processes, 79, 1 (1999), 1-28.

Appendix: Tables and Figures

		Frequency	Valid Percent
Valid	Increase a lot	16	20.3
	Increase somewhat	47	59.5
	Stay the same	16	20.3
	Total	79	100.0
Missing	System	4	
Total		83	

Table 1: Programming Games and Learning Effectiveness

	Frequency	Percent
<18	4	4.8
18-24	29	34.9
25-34	33	39.8
35-49	16	19.3
>50	1	1.2
Total	83	100.0

Table 2: Age of the Developers

			Education	Expertise	Age
Spearman's rho	Enjoy Game	Correlation Coefficient	.033	.076	-.094
		Sig. (2-tailed)	.770	.493	.396
		N	83	83	83

Table 3: Correlation between the motivation to participate in the game and developer's education level, expertise level and age.

Reasons to Participate	Count	Percent case-wide (%)
Fun in programming games	70	87.5
To learn programming	43	53.8
To win the game	26	32.5
To gain peer recognition	13	16.3
To win the prize in contest	9	11.3

Table 4: Motivation factors to engage in the programming games

Reasons	Solve problems	Creativity	Put skills in use	Learn new skills	Feel confident	Gain peer recognition
Very important	65.4%	53.1%	28.8%	22.8%	17.9%	2.7%
Somewhat Important	23.5%	21.0%	37.5%	32.9%	24.4%	6.7%
Neutral	6.2%	16.0%	28.8%	21.5%	33.3%	30.7%
Somewhat Not Important	4.9%	6.2%	3.8%	16.5%	7.7%	24.0%
Not Important	0%	3.7%	1.3%	6.3%	16.7%	36.0%
Mean	4.49	4.14	3.89	3.49	3.19	2.16
Std. Deviation	0.823	1.126	0.914	1.197	1.300	1.079

Table 5: factors that make the programming game engaging

Programming stage	Discover algorithms	Architect implementation	Write the code	Debug and test
Like it very much	52.5%	34.2%	13.9%	3.8%
Somewhat like it	27.5%	36.7%	30.4%	17.9%
Neutral	12.5%	20.3%	40.5%	25.6%
Somewhat dislike it	3.8%	7.6%	10.1%	34.6%
Dislike it very much	3.8%	1.3%	5.1%	17.9%
Mean	4.21	3.95	3.38	2.55
Std. Deviation	1.052	0.986	1.017	1.101

Table 6: Fun in different programming tasks