Just For Fun: Using Programming Games in Software Programming Training and Education — A Field Study of IBM Robocode Community

Ju Long
McCoy School of Business at Texas State University – San Marcos, TX, USA

julong@txstate.edu

Executive Summary

Improving learning effectiveness has always been a constant challenge in software education and training. One of the primary tasks educators face is to motivate learners to perform to their best abilities. Using computer games is one means to encourage learners to learn (Klawe, 1994). When games are used in general education, they could enhance self-esteem for the learners, reduce training time and instructor load, provide more opportunities for practice and enhance knowledge acquisition (Brownfield & Vik, 1983; Ricci, 1994). Moreover, researchers have noted many benefits of using computer games in IT education as well. For instance, it has been shown that a game-based environment is more enjoyable to the users than a traditional environment in IT training (Venkatesh, 1999). In this study, we focused on one game: IBM Robocode. We examined the characteristics of Robocode and the Robocode community. We addressed the following research questions: (1) How did the Robocode game influence the participant's learning outcomes? Did the participant's programming skills and knowledge improve after playing the Robocode game? (2) Was the Robocode game enjoyable to participants with different ages, education and skill levels? (3) What did the participants like the most about the Robocode game? (4) What were the factors that influence the participant's motivation? (5) What activities of the game were more enjoyable? To answer these questions, we conducted a survey of Robocode participants. Our sample was randomly drawn from the online forum of the Robocode community. We sent out a total of 500 surveys and generated 83 valid responses. Regarding the effectiveness in learning new programming skills using Robocode, we found that about 80% of the participants' programming skills increased after playing Robocode. We also found that Robocode was enjoyable not only to younger participants but also to older—and often more experienced—participants. Our results also showed that participants in various education levels and expertise levels all enjoyed Robocode. Examining the motivation for the learners to participate in the game, we found that to have fun (an intrinsic motivator) was the most common reason. Another important factor, also an intrinsic motivator, was the opportunity to learn new programming skills. Extrinsic motivators, such as "to win the game", "to win the prize in the contest" and "to gain peer recognition", proved to be far less significant. Furthermore, we examined the factors that made Robocode fun. Among the factors listed, "to be able to solve problems on my own" was chosen most frequently. "To be able to be creative" was identified as the second most important factor. Both factors are

Material published as part of this publication, either on-line or in print, is copyrighted by the Informing Science Institute. Permission to make digital or paper copy of part or all of these works for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage AND that copies 1) bear this notice in full and 2) give the full citation on the first page. It is permissible to abstract these works so long as credit is given. To copy in all other cases or to republish or to post on a server or to redistribute to lists requires specific permission and payment of a fee. Contact Publisher@InformingScience.org to request redistribution permission.

Editor: Grandon Gill

most closely related to the autonomy category of intrinsic motivation. “To be able to put skills in use” and “to be able to learn new skills” were also selected as important factors. Both factors could be classified as competence motivators. Our results are consistent with the previous theories on motivation that the more autonomy and competence the learner has, the more fun the learning process. We also found that the Robocode participants enjoyed some activities of the game more than other activities. Among the Robocode activities, discovering algorithms was chosen as the most enjoyable. Designing the architecture was highly enjoyable as well. On the other hand, writing the code and testing and debugging were less enjoyable. Our research could provide a better understanding of how to incorporate Robocode and similar games in software programming education. The results from our study could be valuable not only to practitioners, but also to researchers examining the methods to improve software education and training. Instructors hoping to improve learning outcomes could leverage a programming game such as Robocode as an education and training tool to enhance the learning experiences and outcomes. It might also be interesting to broaden the scope and utilize Robocode or similar games in higher-level courses, or even graduate courses.

Keywords: Software Programming Education and Training, Programming Games, Computer Games, Intrinsic Motivation, Robocode.

Introduction

Improving learning effectiveness has always been a constant challenge in software education and training. One of the primary tasks educators face is to motivate learners to perform to their best abilities. One method of motivating learners is the game-based exercise. In this research, we conducted a field study in a community of software participants in the IBM Robocode game. The characteristics of the Robocode game and the Robocode community were examined. Our research could shed some light on how computer-programming games – such as the Robocode game – could be used in software programming education to increase learning effectiveness.

IBM Robocode teaches participants the Java programming language in a game format. It holds tournaments and contests among participants. In our field study, we addressed the following research questions: (1) How did the Robocode game influence the participant’s learning outcomes? Did the participant’s programming skills and knowledge improve after participating in the Robocode game? (2) Was the Robocode game enjoyable to participants with different ages, education backgrounds and skill levels? (3) What did the participants like most about the Robocode game? (4) What were the factors that influence the participant’s motivation? (5) What activities of the game were more enjoyable? In the following section, we review the theories of using computer games in general education and in software programming education. In the third section, we discuss our field study and analyzed the research results. The final section presents our conclusions and directions for future research.

Using Computer Games in General Education

Computer games are seen as a means to encourage learners who may lack interest to learn (Klawe, 1994). Games have also been used in enhancing self-esteem for the learners who may lack confidence in learning (Dempsey, Rasmussen, & Lucassen, 1994; Ritchie & Dodge, 1992). When games are used in training and educational settings, it is suggested that they can reduce training time and instructor load, providing more opportunities for practice and enhancing knowledge acquisition (Brownfield & Vik, 1983; Ricci, 1994). Computer games also lead to positive results in long-term learner retention by improving learning interests and more focused attention, because the students enjoy the approach (Randel, Morris, Wetzel, & Whitehill, 1992; Ricci, 1994). Games have been used to encourage learning particularly in curriculum areas such as math, physics and language arts, where specific objectives can be stated (Randel et al., 1992).

Games have also been used in training the vision of partially sighted children (Sik-Lányi & Lányi, 2003), and teaching labeled transition systems (Roussev, 2003).

Computer games can also foster understanding of theoretical models and interaction effects and can support the development of team, social, communication and resource sharing skills (Berson, 1996; Helliar, Michaelson, Power, & Sinclair, 2000; Hollins, 2003; Ritchie & Dodge, 1992; Squire et al., 2003). Since building team skills and communication skills are important components in software programming education, incorporating computer games could be beneficial.

Researchers have also studied why computer games could improve 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 (Prensky, 2001). Moreover, computer games are flexible and complex enough to cater for different learning styles (Kirriemuir, 2002; Sedighian, 1994). They can deal with infinite amounts of contents and afford differing levels of challenges. The instant feedback and risk-free environment of computer games invite exploration and experimentation, stimulating 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 Software Programming Education

Research has explored how computer games could be used specifically in IT education. For instance, researchers have studied how a micro-world (a learner-centered world that is explored by directly manipulating objects in the world with a limited set of simple commands coupled with metaphors to aid in problem description and to exploit storytelling) could be used as an educational paradigm (Kelleher & Pausch, 2005). Examples of micro-worlds include Karel the Robot (Pattis, 1995) and Alice (Alice.org, 2005).

Researchers have noted many benefits of using computer games in IT education. For example, an empirical study compared a traditional training environment with a game-based environment, with the latter being constructed so as to be more enjoyable to the users (Venkatesh, 1999). It was found that ease-of-use perceptions were higher with the game-based training group than with the traditional training group. Another study suggested that users of game-based training approached technology with a more imaginative style that encouraged skill development (Martocchio & Webster, 1992). Further research also found that game-based training led to higher levels of interest in further information (Sansone, Sachau, & Weir, 1989). Researchers have also used a simulation game of an inventory system in teaching operations management. They found that the game helped students understand concepts more quickly and remember them better than with a lecture (Klassen & Willoughby, 2003).

A particularly interesting programming game is IBM Robocode. Created by Mat Nelson, Robocode was designed to promote the learning of Java. Participants in Robocode community are real-world software developers, albeit with various levels of expertise and experiences. To play a Robocode game, each developer needs to create a robot program using Java. The Robocode framework defines the basic physical rules every robot has to follow and provides a re-usable object structure to ease the development. Participants 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.

As a Java-based environment, Robocode provides a well-defined domain for students to learn and apply concepts of object-oriented design and programming, such as proper design of classes, extension and re-use of existing codes using inheritance mechanisms, event handling and message passing (Bonakdarian & White, 2004). Robocode has been implemented in the classroom to stimulate student learning-outcomes. One study found that when Robocode was utilized in the classroom, students really enjoyed the project and also produced some very creative solutions (Bierre, Ventura, Phelps & Egert, 2006). Research also found that Robocode enabled students to develop skills for each stage of the software development process and fostered critical thinking (O'Kelly & Gibson, 2006). Moreover, the results of the student's efforts in Robocode can be seen instantly. Such instant gratification can speed up the learning cycle and is analogous to the "prototyping model" used in software development (O'Kelly & Gibson, 2006). Robocode also forces students to consider issues of "computer intelligence", i.e., how do agents sense and react to their environment to maximize their goals? Thus, it is useful for courses related to artificial intelligence (Bonakdarian & White, 2004). By incorporating an enjoyable game into an artificial intelligence class and providing students with tools for developing practical versions of the algorithms, students appreciated the theory better and developed greater confidence in their understanding of it (Hartness, 2004).

The research reviewed above provided valuable insights into the benefits of using Robocode in programming courses. Some questions, however, still remained largely unexplored. For example, was Robocode effective across all age groups, educational backgrounds, and experience levels? Were all the stages of Robocode equally enjoyable to the participants? What were the factors that made Robocode enjoyable to the participants? A better understanding of these questions could provide researchers and educators with some insights on Robocode's implementations in the classroom. That was the aim of our field study.

Research Methods and Data Collection

We selected Robocode as our research site because of its educational value to the participants. To play Robocode effectively, participants must acquire a significant amount of expertise to continuously create, maintain and enhance their programs. During the process, participants learn Java language in every aspect of the software development process, including algorithm design, architecture, implementation, optimization, testing and bug fixing. From a research perspective, Robocode provided us three distinctive benefits:

1. Unlike simulations conducted in classroom settings, Robocode is a real-world programming game with real world participants, increasing the validity of our research.
2. Its diverse community ensured that the subjects in our sample would have various programming skills and education levels.
3. The participants all worked on the same software on the same platform using the same programming language. This homogenous developing environment ensured that the development tasks and processes were similar for all participants. Thus, it reduced the degree to which different programming languages or programming environments could act as confounding variables.

A survey of Robocode participants was our main data collection method. The Robocode community maintains a very active online discussion forum. Our survey sample was randomly drawn from the forum. We sent out a total of 500 surveys and generated 83 valid responses excluding two invalid responses (A response rate of 17%).

Data Analyses and Results

Robocode Could Effectively Improve Learning Outcomes.

Our survey results suggested that the Robocode game was a very effective tool for promoting self-motivated learning—considered by many to be the best way to learn (Lepper & Malone, 1987). One survey question specifically addressed perceptions of effectiveness in learning new programming skills using Robocode. About 80% of the participants reported that their programming skills increased through participating in Robocode. Among these participants, more than 20% said that their skills had improved significantly and about 60% reported that their skills had increased to some extent. Only 20% reported that their skills stayed about the same (see Table 1). This result strongly suggested Robocode was effective in promoting self-motivated learning.

Table 1: Programming games and learning effectiveness

		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	

Robocode and the Age, Educational and Expertise Levels

Participants in our sample came from all age groups (see Table 2). The majority of them (75%) were between the ages of 18 and 34, with about 40% of the participants in the 25 to 34 age group, and another 20% in the 35 to 49 age group. The fact that the majority of the participants (60%) were more than 25 years old suggested that Robocode was enjoyable not only to younger participants but also to older—and often more experienced—participants. A correlation test between the participants' time spent playing the game and their age groups did not result in any significant correlation (see Table 3). This result further demonstrated that participants of various age groups enjoyed Robocode. Notably, our findings were consistent with the results of a previous study that examined game-based versus traditional training, and did not find evidence of a moderating effect of age on the effects of the training method (Venkatesh, 1999).

Table 2: Age of the participants

	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 3: Correlation between the time spent in playing the game and developer's education level, expertise level and age

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

The education levels of the participants were quite high, which presented another strong contrast to the common belief that games are more enjoyable to population at the high school or college education level. In our sample, 48.2% of the participants had some graduate school education or a graduate degree, with another 42.2% having some college education or an undergraduate degree. This finding implied that Robocode could be enjoyable to participants across various education levels. To further confirm this finding, we conducted a correlation test on the education level and the time spent in playing the game. We did not find a significant correlation between these two variables (see Table 3). This result further supported the finding that participants in various education levels enjoyed Robocode.

When joining Robocode, participants must assign themselves into one of those three leagues based on their programming skills. The beginners consist of participants with no previous Java programming experience. The intermediate participants are those with some experience (less than six months) in Java or substantial experience with another programming language; the advanced participants are experienced and skilled in Java. Using the same criteria, we asked the participants to rate their skill levels in the survey. The results showed that our sample included participants from all three levels: approximately 23% of the participants were beginners, approximately 40% were intermediate participants, and about 37% were advanced participants. We conducted a correlation test on the participant's time spent in the game with their expertise level and did not find significant correlation.

In summary, our findings suggested that time spent in playing the Robocode is not influenced by the participants' expertise levels, age or education backgrounds.

Factors that Influence the Learner's Efforts in Robocode

One of our main research questions was to study the factors that kept the learners playing Robocode. A clear understanding of these factors could help us develop strategies to keep participants engaged in the game over the long term. Based on our preliminary pre-test, we included the following factors in the survey:

- 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
- to gain recognition among peers (see Table 4).

Table 4: Motivation factors to engage in Robocode

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

We then mapped these factors into the intrinsic and extrinsic motivation categories of the motivation theory (Deci, 1975), as shown in the model framework (see Figure 1).

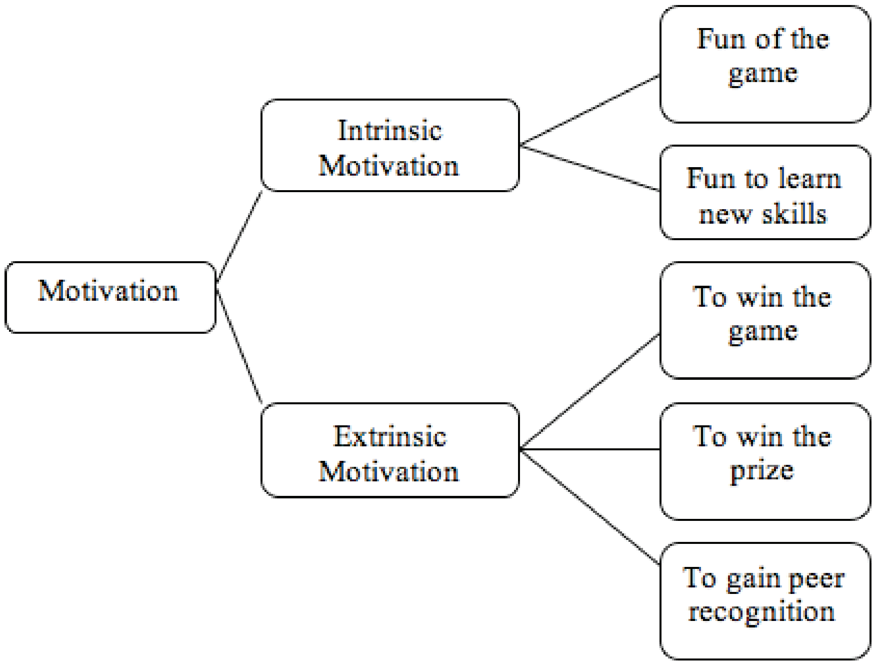


Figure 1: Motivational Factors to Participate in Robocode

Our survey results found that to have fun (an intrinsic motivator) was the most common reason for participating in the game. 87.5% of participants identified fun of game as one of their motivations. Another important factor, also an intrinsic motivator, was the opportunity to learn new programming skills, identified by 54% of the participants. Extrinsic motivators, such as "to win the game", "to win the prize in the contest" and "to gain peer recognition", proved to be far less significant. For example, 32.5% of the participants chose "to win the competition" as one of the motivations. Even fewer participants (11.3%) chose "to win the prize" as a motivation. "To gain recognition among the peers" did not appear to be important to most players either (only 16.3% participants chose it).

We conducted a regression analysis to study how these motivational factors influenced the participants' efforts to learn. Because the participants' effort level is a latent variable, we used the number of hours the participants spent in the game as a manifest indicator. Our results suggested that most of the participants were very active. Nearly 60% of the participants spent more than 1 hour each day playing Robocode, with 16% spent more than 4 hours each day (see Table 6).

The following is the regression model:

Learning Effort = $\alpha + \beta_1 Fun + \beta_2 Learn + \beta_3 Peer + \varepsilon$

All three motivators were significant at the 95% level in our model (see Table 5).

Table 5: Regression analysis of programming effort and motivation factors

ANOVA

Model		Sum of Squares	Df	Mean Square	F	Sig.
1	Regression	20.369	3	6.790	6.426	.001(a)
	Residual	82.412	78	1.057		
	Total	102.780	81			

a Predictors: (Constant), To gain recognition of the peers, Programming game is fun, To learn new programming skills

b Dependent Variable: Efforts in the game

Coefficients (a)

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std. Error	Beta		
1	(Constant)	1.080	.320		3.375	.001
	Fun	.626	.313	.204	1.999	.049
	Learn New Skills	.778	.229	.347	3.393	.001
	Gain Peer Recognition	.735	.314	.240	2.341	.022

a Dependent Variable: Efforts in the game

Why Robocode is Enjoyable?

We examined the factors that made Robocode fun. Among the factors listed, "to be able to solve problems on my own" was chosen most frequently. 65.4% of the participants identified it as very important, and 22.9% of the participants chose it as important. "To be able to be creative" was identified as the second most important factor, selected by 53.1% participants as being very important and by another 21% participants as being important. Both of these factors are most closely related to the autonomy category of intrinsic motivation (see Table 6 and Table 7).

Table 6: Number of hours spent in playing the game

	Frequency	Valid Percent	Cumulative Percent
less than 1 hour	34	41.5	41.5
1-2 hour	17	20.7	62.2
2-4 hour	18	22.0	84.1
more than 4 hours	13	15.9	100.0
Total	82	100.0	

Table 7: Factors that make Robocode engaging

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

Of the next two factors in overall importance, “to be able to put skills in use” was selected by 65.3% of the participants and “to be able to learn new skills” was selected by 55.7% of the participants. Both of these factors could be classified as competence motivators.

Our results are consistent with the previous theories on motivation that the more autonomy and competence the learner has, the more fun the learning process, and the more motivated the learners (Deci, 1975). This result is illustrated in the framework shown in Figure 2.

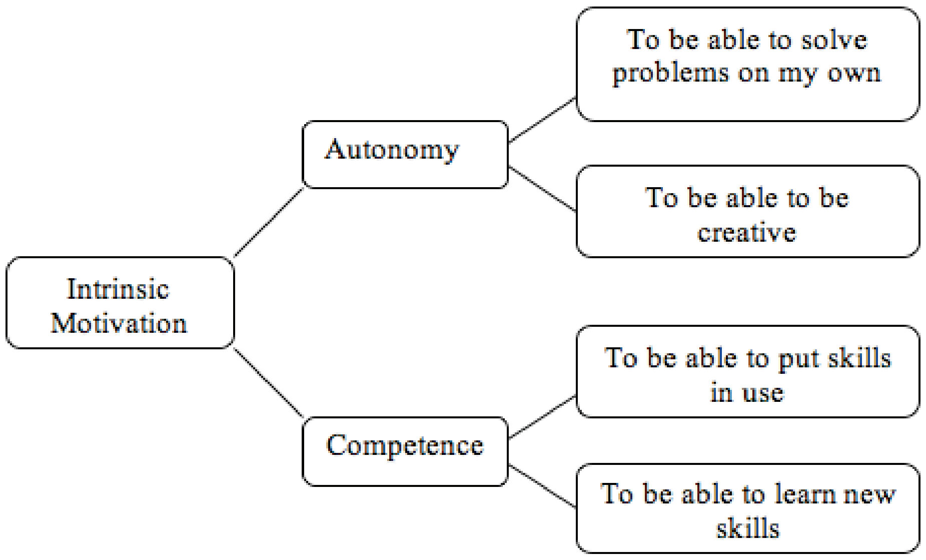

To be able to solve problems on my own
Autonomy
To be able to be creative
Intrinsic Motivation
To be able to put skills in use
Competence
To be able to learn new skills

Figure 2: Factors that influence participants’ intrinsic motivation

Some Programming Activities Are More Enjoyable to the Participants

We also examined whether the various activities of playing Robocode were equally enjoyable to the participants. Based on our exploratory pre-test results, we divided the programming activities into the following categories:

1. discovering the algorithms,
2. designing the architecture,
3. writing the code, and
4. testing and debugging (see Table 8).

Among the above activities, discovering algorithms was chosen as the most enjoyable, with more than 80% of participants enjoying this task (52.5% participants liked it very much, and 27.5% somewhat liked it). Similar to algorithm design, designing the architecture was found to be highly enjoyable as well (34.2% participants liked it very much, and 36.7% somewhat liked it). Both discovering algorithms and designing architectures are inherently creative processes that offer considerable autonomy. Therefore, this result was consistent with the previously reported findings that players are attracted to the autonomy-based aspects of Robocode.

Compared with more creative tasks of discovering algorithms and designing the architecture, writing the code and testing and debugging were less enjoyable. Only 44.3% of the participants enjoyed writing code; and only 21.7% of the respondents enjoyed testing and debugging (while 78.2% did not like testing or debugging or were neutral about it). It could be argued that these activities offer relatively low levels of autonomy and competence. Creativity, in particular, tends to be relatively low in coding and debugging activities, since writing code is relatively mechanical once the algorithms have been designed.

Table 8: Fun in different programming tasks

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

Research Implications and Conclusion

A better understanding of how to incorporate games in software programming education would be valuable not only to practitioners, but also to researchers examining the methods to improve software education and training. Based on our field study of Robocode, we confirmed that Robocode could improve the participant's motivation in the learning process and achieve better learning effectiveness. We showed that Robocode could be applied to learners with various skills, experiences and ages. We also found that some programming tasks, such as discovering the algorithm, are more enjoyable to participants than other tasks, such as debugging and testing.

Several of the findings from our study could be applied when educators implement Robocode and similar games in programming courses. For example:

- Because the enjoyment in Robocode appeared to stem from intrinsic motivators, particularly autonomy, instructors could try to provide students with substantial opportunities for creativity in designing programming instructions.
- Because participants across various age groups, expertise levels and education backgrounds all enjoyed Robocode, it would appear that findings from existing research regarding the effectiveness of using Robocode in introductory programming course are likely to generalize well to the use of Robocode or similar games in higher-level courses, or even graduate courses.
- Since some development tasks were more enjoyable to participants than other tasks, project teams could be constructed with a balance of skills and preferences so that different individuals may be assigned work at their preferred tasks (O'Kelly & Gibson, 2006).

It is also important that games are used to facilitate tasks appropriate to learners' level of maturity in the skill (Din & 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 gratifications so as to increase the sense of autonomy and competence. The designers could vary between short modules (to maximize the likelihood of satisfactory outcomes) but also make available longer sessions (to encourage involvement). Furthermore, the game should be able to provide different levels of challenges and cater to different learning levels. The Robocode league system could serve as an excellent model for such an implementation.

Because our study was designed as an explanatory field study instead of a controlled experiment, our results were exclusively based on the participant survey. In our future research, we plan to embed subjects in the Robocode game and test their learning outcomes using the controlled field experiment method. In addition, it is interesting to note that all the participants in our current study were male. This may be partly because Robocode was not a game appealing to women participants. Thus the results from our sample could only represent the experiences and characteristics of male participants.

References

Alice.org (2005). http://www.alice.org accessed on September, 06.

Berson, M.J. (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, J.A. (1995). Computer games: Increase learning in an interactive multidisciplinary environment. *Journal of Educational Technology Systems*, *24*(2), 195–205.

Bierre, K., Ventura, P., Phelps, A. & Egert, C. (2006). Motivating OOP by blowing things up: An exercise in cooperation and competition in an introductory java-programming course. *ACM SIGCSE Bulletin, Proceedings of the 37th SIGCSE technical symposium on Computer science education SIGCSE '06*, 38(1).

Bonakdarian, E. & White, L. (2004). Robocode throughout the curriculum. *Journal of Computing Sciences in Colleges*, *19*(3).

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

Deci, E. L. (1975). *Intrinsic motivation*. New York: Plenum Press.

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

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

Hartness, K. (2004). Robocode: Using games to teach artificial intelligence. *Journal of Computing Sciences in Colleges*, *19*(4).

Helliar, C., Michaelson, R., Power, D., & Sinclair, C. (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.

Kelleher, C. & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR), 37*(2).

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*. Retrieved from www.jisc.ac.uk/index.cfm?name=techwatch_report_0201

Klassen, K. & Willoughby, K. (2003). In-class simulation games: Assessing student learning. *Journal of Information Technology Education, 2,* 1-13. Available at http://jite.org/documents/Vol2/v2p001-013-59.pdf

Klawe, M. (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., & Malone, T.W. (1987). Intrinsic motivation and instructional effectiveness in computer-based education. In R. E. Snow& M. J. Farr (Eds.)*, Aptitude, learning and instruction* (pp. 255-286). Hillsdale, NJ: Erlbaum.

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

O'Kelly, J. & Gibson, P. (2006). RoboCode & problem-based learning: A non-prescriptive approach to teaching programming. *ACM SIGCSE Bulletin, Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education ITICSE '06*, 38(3).

Pattis, R. E. (1995). *Karel the robot: A gentle introduction to the art of programming* (2nd ed.). John Wiley & Sons.

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

Randel, J.M., Morris, B.A., Wetzel, C.D., & Whitehill, B.V. (1992). The effectiveness of games for educational purposes: A review of recent research. *Simulation and Gaming*, *23*(3), 261–276.

Ricci, K.E. (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.

Roussev, B. (2003). Teaching Introduction to Programming as Part of the IS Component of the Business Curriculum. *Journal of Information Technology Education, 2*, 349-356*.* Available at http://jite.org/documents/Vol2/v2p349-356-43.pdf

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

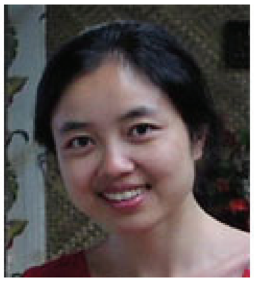
Sedighian K (1994). Playing styles for computer and video games. In T. Ottman & I. Tomek (Eds.), *Proceedings of the ED-MEDIA 94 World Conference on Educational Multimedia and Hypermedia. Panel discussion 'Can electronic games make a positive contribution to the learning of mathematics and science in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

Sik-Lányi, C. & Lányi, Z. (2003). Multimedia program for training of vision of children with visual impairment and amblyopia. *Journal of Information Technology Education, 2*, 279-290. Available at http://jite.org/documents/Vol2/v2p279-290-28.pdf

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

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

Biography



Ju Long is an assistant professor of Computer Information Systems at the McCoy School of Business at Texas State University –San Marcos. She has a PhD. degree from the University of Texas at Austin and a master degree from the University of Michigan in Ann Arbor. She studies technology's impact on business and how new software development methodologies are changing our world.