A Game Based, Financial Literacy Oriented Approach to Improving Programming Education

Hongwei Zhu
Information Technology & Decision Sciences, Old Dominion University, Norfolk, VA, United States., hzhu@odu.edu

Yuzhong Shen
Modeling, Simulation and Visualization Engineering, Old Dominion University, Norfolk, VA, United States., yshen@odu.edu

A Game Based, Financial Literacy Oriented Approach to Improving Programming Education

Hongwei Zhu
Old Dominion University
hzhu@odu.edu

Yuzhong Shen
Old Dominion University
yshen@odu.edu

ABSTRACT

Every year, two thirds of college seniors (about 1.8 million) in the US graduate with significant debts, but most of them are poorly equipped with essential financial knowledge to manage their debts and make intelligent financial decisions. Programming courses are uniquely positioned to offer opportunities to help students improve financial literacy. However, there have been no integrated courses to exploit the synergy. Meanwhile, computing disciplines face continued challenges of getting students interested in computing and finding ways to improve learning effectiveness. To address these challenges, we are developing an innovative teaching strategy that infuses financial literacy into four computing courses and engages students to develop financial literacy games. Students' interests and learning outcomes will be improved because they enjoy computer games and are motivated when they use computing skills to address issues closely related to their daily lives.

Keywords

Computing education, programming, computer games, game-based learning, financial literacy.

INTRODUCTION

Programming is an essential component of the curricula of all five sub-disciplines of computing, which include Computer Science, Computer Engineering, Information Systems, Information Technology, and Software Engineering (ACM AIS IEEE-CS 2009). Programming courses are offered in most of 4,000 colleges and universities in the country, and the need for trained workers continues to be high. However, computing disciplines face declining enrollment as student interest appears to have waned. The world of programming, replete with data types, inheritance, software design patterns, and other important programming may hold little relevance for students whose world is struggling with the housing bubble, financial crisis, job losses, and mounting debts. Debt weighs heavily on today's college students, and more than two thirds of college students (i.e., 1.8 million) graduate with significant student loan obligations. Without the financial knowledge needed to effectively manage debt, students may even regret the choice to go to college. A recent graduate with $60,000 in student loans commented, "Sometimes I think going to school is the worst single mistake I've made" (Damast 2009). The need for financial literacy for college students is critical and provides an opportunity for a novel approach to computer programming education. To better engage students in learning essential programming skills, we must change *how* we teach programming, as well as *what* we teach. We must relate the curricula to the lives of students. Furthermore, students usually learn well with hands-on tools such as computer games. A few online games teach certain aspects of personal finance, but lack realistic scenarios and are too simplistic to be effective.

We have designed teaching strategy that directly addresses these challenges in programming education. In this paper, we describe this strategy. The approach is being implemented and additional results will be incorporated into the revision of the paper at the end of the April.

OVERVIEW OF APPROACH

We have designed the approach to unite programming and financial literacy in five courses: four programming courses and one personal financial literacy course. This approach has three major components:

1. Students in selected programming courses will develop software components and libraries such as loan calculators and investment management tools;
2. In a game development course, students will integrate these software components into a serious financial literacy game;

3. The financial literacy game will be employed in related computer courses and in a university-wide general education course on personal financial literacy.

Figure 1 shows the relationships of the courses and components as well as the implementation timeline. Table 1 provides detailed information about the courses involved.

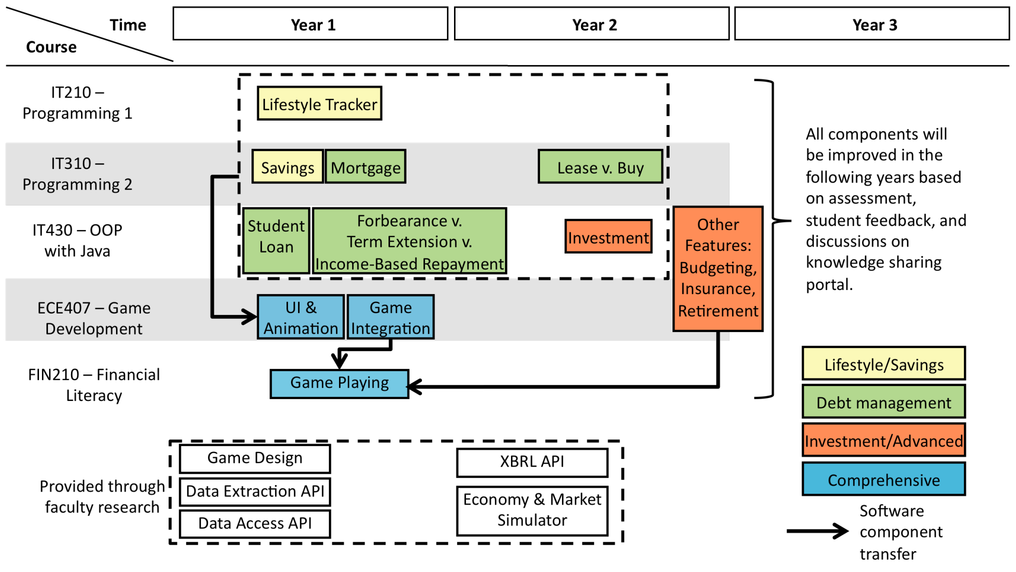

Time
Course
Year 1
Year 2
Year 3
IT210 – Programming 1
IT310 – Programming 2
IT430 – OOP with Java
ECE407 – Game Development
FIN210 – Financial Literacy
Lifestyle Tracker
Savings
Mortgage
Lease v. Buy
Student Loan
Forbearance v. Term Extension v. Income-Based Repayment
Investment
Other Features: Budgeting, Insurance, Retirement
UI & Animation
Game Integration
Game Playing
All components will be improved in the following years based on assessment, student feedback, and discussions on knowledge sharing portal.
Lifestyle/Savings
Debt management
Investment/Advanced
Comprehensive
Software component transfer
Provided through faculty research
Game Design
Data Extraction API
Data Access API
XBRL API
Economy & Market Simulator

Figure 1. Courses and components of the approach and implementation timeline.

Course	Curriculum Requirement	Topics and Objectives
IT210 – Programming 1	Required, IT major	Intro to programming, to provide students with an understanding of the C++ language elements and an opportunity to apply the language in a practical context.
IT310 – Programming 2	Required, IT major	Topics include OO concepts and GUI business applications. To prepare students with problem solving skills.
IT430 – OOP with Java	Elective, IT & CS majors	Advanced topics in OOP, software design patterns, and software engineering principles, which are reinforced by implementing practical business applications.
ECE407 – Game Development	Elective, ECE & CS majors	Topics include game architecture, computer graphics theory, user interaction, audio, high level shading language, animation, physics, and artificial intelligence. Students develop games related to STEM education.
FIN210 – Personal Financial Literacy	General education	Introduction to various aspects of individual financial decision making, with an emphasis on short- and long-term personal financial planning. It uses scenarios, practical cases, and special projects to provide concrete applications of abstract concepts.

Table 1. Characteristics of Courses

Based on information about student loans garnered from Department of Education statistics, the game will focus on three areas of financial literary: (1) Lifestyle and savings; (2) debt management; and (3) investments and advanced topics. Infusing these personal finance concepts into programming courses will allow for practice of computing skills, but will also help students understand the importance of saving money, of carefully managing student loans and other forms of debt, as well as the basics of investment. The components of the focus areas are color coded in Figure 1. The purposes and programming knowledge involved are summarized in Table 2.

Component	Functions/Purposes	Programming
Lifestyle Tracker	Keeps track of lifestyle choices, including anything from entertainment spending (e.g., going to parties, movies, concerts, etc.), purchasing habits, and money saving habits. Profiles are created based on the choices to estimate expenses and savings (in conjunction with the *Savings* component) of players over the simulated time period.	Data types, data operations, and record keeping
Savings	Uses interest rates found online to compute account balances; uses current and projected information rates to compute real rate of return.	OO methodology, use of data extraction API
Mortgage	Various mortgage calculations such as affordability, amortization, savings from early payment, fixed rate vs. adjusted rate.	OO methodology, GUI
Lease v. Buy	Compares leasing v. buying a car. Similarly, rent v. own can be compared for real estate. Data about current market are obtained online.	OOP, software design pattern
Student Loan	Finds and compares loan offers; calculates pay-off date based on expected income and chosen lifestyle.	OOP, software design pattern, data extraction
Forbearance v. Other Choices	Compares different student loan payment choices based on expected income and chosen lifestyle.	OOP, software design pattern
Investment and Advanced Features	Investment choices based on preferred risk level; analyze company financial in XBRL; effects of tax deferral on return; retirement planning.	OOP, system development and code reuse, XML processing
Game Implementation	Design game storyline, implement game features by integrating components, design and implement human-computer interactions (HCI).	OOP, application integration, computer graphics, HCI

Table 2. Incorporating Financial Literacy into Programming Education

This approach will allow students to see firsthand how to build software components for complex software systems that solve problems closely related to their daily lives. Furthermore, the use of computer games will attract more students to computing disciplines while improving learning effectiveness. Multiple game versions will emerge from the students' creative work. (Here, *games* refers to multiple versions of the game; the terms game and games are used interchangeably in the rest of the paper.)

JUSTIFICATION

In this section, we provide discussions and evidence to demonstrate the expected effectiveness of the approach.

Rationale

One of the chief problems facing college students today is student loan debt. The average loan amount of graduating seniors has reached $22,500 (Department of Education 2009), and the total for outstanding student loans in the U.S. has surpassed $675 billion in the nation (Damast 2009). The current economic downturn, along with the challenge of finding employment has resulted in a soaring student loan default rate. Faced with high debt, students and recent graduates are expressing doubt regarding the value of college education. "You often hear the quote that you can't put a price on ignorance. But with the way higher education is going, ignorance is looking more and more affordable every day," said a student graduated a year ago with $29,000 in student debt (Bernard 2009). Those who go to college will be more likely to rely on heavy borrowing (Ramírez 2009).

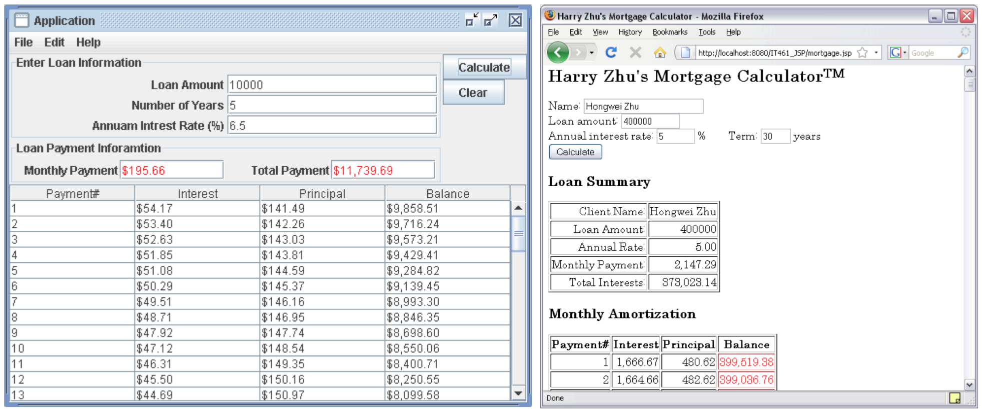
Unfortunately, most college students lack the financial literacy necessary to make sound financial decisions and effectively manage their debts. College education offers students little exposure to personal finance topics, and courses offered on personal finance are usually electives that target business and management majors. As a result, almost two million students graduate each year, poorly equipped to deal with loan obligations. Personal finance management involves information processing, data gathering and number crunching to compare various what-if scenarios. These tasks are a perfect fit for programming courses as students can practice their skills by developing applications for personal finance management. However, no such integrated curricula exist to date. Furthermore, students usually learn well with hands-on tools such as computer games. A few online games teach certain aspects of personal finance, but lack realistic scenarios and are too simplistic to be effective.

Despite many improvements made in programming education (Pears et al. 2007; Robins et al. 2003), little has been done to exploit the synergy between programming and finance in undergraduate education due to disciplinary barriers. Typical programming textbooks and courses use many example and applications to illustrate the key programming concepts. But these examples and applications are often designed with the "concepts" in mind, not with the "students" or the issues concerning the students in mind. Therefore they often change from topic to topic, as if they were created at random. As a result, they are not effective in helping students learn programming. Even worse, students lose interests in learning programming when they do not see how they can use what they learn to solve the problems they face. Most today's students enjoy playing computer games. We should leverage student interests in games by engaging them in game development. This will give students hands-on experiences of using their programming skills and applying software engineering principles.

Results from Prior Work

The effectiveness of the teaching strategy is supported by the results of our prior work that incorporated financial literacy into IT 430, *Objected-Oriented Programming (OOP) with Java* (Zhu 2011), and engaged students in extensive programming tasks in ECE 407, *Introduction to Game Development* (Shen 2009a; Shen 2009b; Shen 2010).

In teaching IT 430, we incorporated financial topics in the forms of example programs and programming assignments throughout the course. Despite the "theme" of financial applications, only three financial programs are used to avoid over-burdening students with financial topics so that they can focus on programming. The programs were implemented multiple times using different methods to help students learn essential programming skills and software engineering principles, ranging from language constructs to OOP concepts and software design patterns. Figure 2 shows the screenshots of a mortgage calculator implemented as a standalone application using the Model-View-Controller (MVC) design pattern and as a Web application that reuses the Model component of the MVC implementation.


Application
File Edit Help
Calculate
Clear
Harry Zhu's Mortgage Calculator - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Google
Done

Enter Loan Information

Loan Amount: 10000
Number of Years: 5
Annuam Intrest Rate (%): 6.5

Loan Payment Inforamtion

Monthly Payment: $195.66
Total Payment: $11,739.69

Payment#	Interest	Principal	Balance
1	$54.17	$141.49	$9,858.51
2	$53.40	$142.26	$9,716.24
3	$52.63	$143.03	$9,573.21
4	$51.85	$143.81	$9,429.41
5	$51.08	$144.59	$9,284.82
6	$50.29	$145.37	$9,139.45
7	$49.51	$146.16	$8,993.30
8	$48.71	$146.95	$8,846.35
9	$47.92	$147.74	$8,698.60
10	$47.12	$148.54	$8,550.06
11	$46.31	$149.35	$8,400.71
12	$45.50	$150.16	$8,250.55
13	$44.69	$150.97	$8,099.58

http://localhost:8080/IT461_JSP/mortgage.jsp

Harry Zhu's Mortgage Calculator™

Name: Hongwei Zhu
Loan amount: 400000
Annual interest rate: 5 % Term: 30 years
Calculate

Loan Summary

Client Name:	Hongwei Zhu
Loan Amount:	400000
Annual Rate:	5.00
Monthly Payment:	2,147.29
Total Interests:	373,023.14

Monthly Amortization

Payment#	Interest	Principal	Balance
1	1,666.67	480.62	399,519.38
2	1,664.66	482.62	399,036.76

(1) GUI application implemented using MVC design pattern (2) Web application reusing Model component

Figure 2. Mortgage calculator implemented as a GUI application and a Web application.

Table 3 shows the results of a survey of the students enrolled in Dr. Zhu's Spring 2009 course. The approach was effective in helping students learn programming and financial literacy simultaneously. (Note: the percentages of "Neither Yes or No" responses are not shown.)

Question	Yes	No
1. I have a student loan	71%	29%
2. Mortgage calculator helped me learn programming	79%	0%
3. Mortgage calculator helped me understand debt	54%	33%
4. I am interested in learning to write programs that will help me manage and reduce debt	79%	4%
5. I think computer tools can help me reduce expenses	67%	4%
6. I think computer tools can help me make better financial decisions	79%	4%
7. I invest or plan to invest my money	88%	0%
8. I can read and analyze company financial reports	46%	17%
9. I am interested in using my programming skills to write programs to gather and process financial info.	46%	17%

Table 3. Survey results (Spring 2009, N=24)

A majority of students had student loans, and most found the various implementations of a mortgage calculator helped them learn both programming and debt management. Other answers indicate that students generally have positive attitudes towards using programming skills to address financial issues.

The effectiveness of the approach was also reflected in student grades. The grade point average for the course had increased from 2.4 (2006) to 3.4 (2009) since the adoption of the integrated teaching strategy. Additional details of the experimental approach can be found in (Zhu 2011).

Since 2009, we have been offering ECE 40 to teach game development theory and techniques with emphasis on education game development (Shen 2009a; Shen 2009b; Shen 2010). It quickly became popular. Course topics include game architecture, 2D and 3D games, computer graphics theory, user interactions, audio, high level shading language, animation, content generation, game deployment, mathematics, physics, and artificial intelligence. Students develop games for Science, Technology, Engineering, and Mathematics (STEM) education at different levels, and these games can be deployed on multiple hardware platforms, including PCs, game consoles and mobile platforms. In addition, the games can be played locally or through the network. Students enrolled in this course have come from different majors, including computer science, modeling and simulation, computer engineering, electrical engineering, information technology, and mechanical engineering. This course is an important component of Old Dominion University's game curriculum, which was selected by The Princeton Review as one of the top 50 undergraduate game design programs in the US.

Course evaluations were conducted at the end of the Spring 2009 and Spring 2010 semesters and the overall feedback from the students was very positive. The evaluation results indicated that the lectures and programming assignments were well structured and very interesting, that the course improved students' object-oriented design and programming skills, and that, overall, they benefited from this course.

Theoretical Foundation and Empirical Support

Programming assignments and course projects will provide students with opportunities to integrate multidisciplinary knowledge and solve practical financial problems. This hands-on, learning-by-doing practice will elevate student interest and improve learning effectiveness (Davis 1993; Felder and Peretti 1998; Starrett and Morcos 2001).

Electronic games, including video and computer games, are a pervasive aspect of American culture and entertainment: as many as 65 percent of American households play games (Entertainment Software Association 2009). The game industry has evolved into an important business sector that is larger than the film industry with an annual revenue of $20.2 billion in 2009 (NPD Group 2010). This passion for games can be exploited for more vital purposes, such as education, training, and marketing, via “serious games.” Game-based learning uses serious or educational games with defined learning outcomes and objectives. Traditionally game-based learning mainly targeted young children, but all ages can benefit from game-based learning. A number of commercial and free game-based learning software packages have been released, including *Reader Rabbit*, a series of software titles for kids aging from toddlers to 3rd graders that teaches learn math and reading (The Learning Company 2009); *Fun Brain*, a very popular website of Pearson Education, which provides many games on math and reading (Pearson Education Inc. 2009); and *Alice*, a 3D graphical programming environment for generating interactive animations and games (Cargenie Mellon University 2009a) that has been used by over 1,300 schools to teach both game development and introductory object-oriented programming (Cargenie Mellon University 2009b). Football games, such as *Madden NFL*, have been routinely used by NFL teams for planning and training purposes, as *Microsoft Flight Simulator X* is for flight simulations. *Whyville* is a popular virtual world for children to learn various subjects such as science and geography and to interact with each other (Numedeon Inc 2009). Research has even found that surgeons who played video games performed better in surgeries. Finally, the value of game-based learning has been recognized by national organizations such as National Science Foundation and National Research Council (American Association for the Advancement of Science 2005; Project Kaleidoscope 2002; The National Academies 2003).

IMPLEMENTATION

We are currently in Year 1 of the implementation timeline discussed earlier. We have completed a preliminary design of the game, which we call “My Life”. Figure 3 is shows a schematic view of the architecture of core components of the game engine. Player profile will be initially obtained via a question-answer session and continuous adjusted according to player decisions made during game play. Certain events that affect student life will be generated during game play.

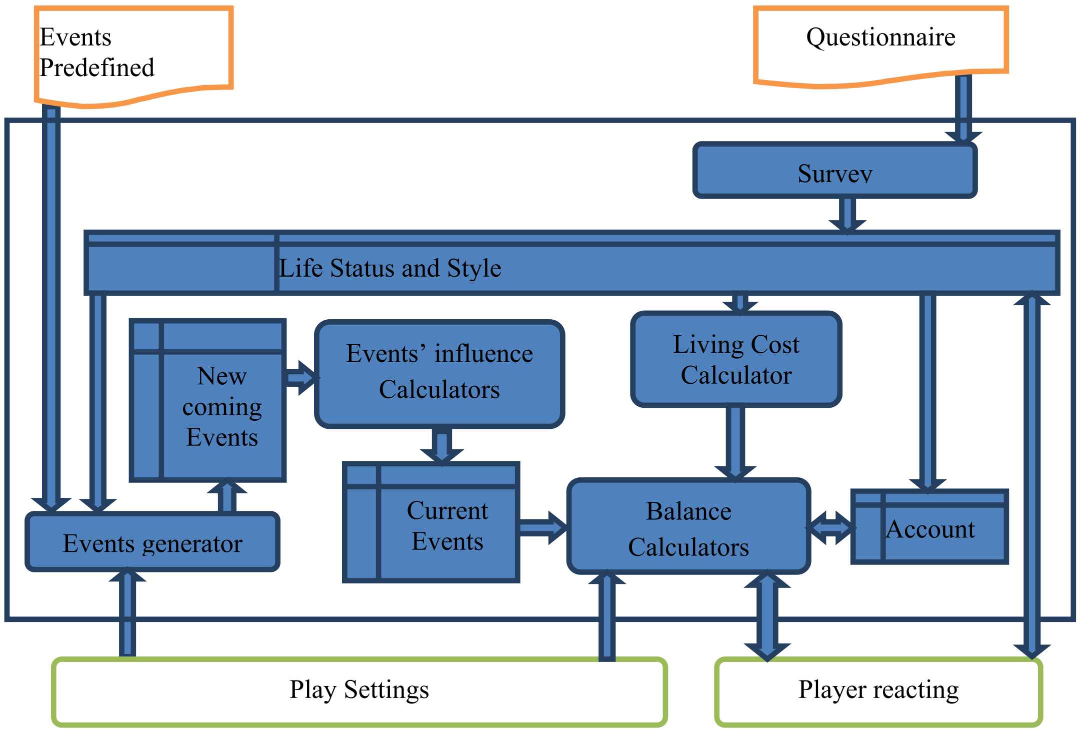
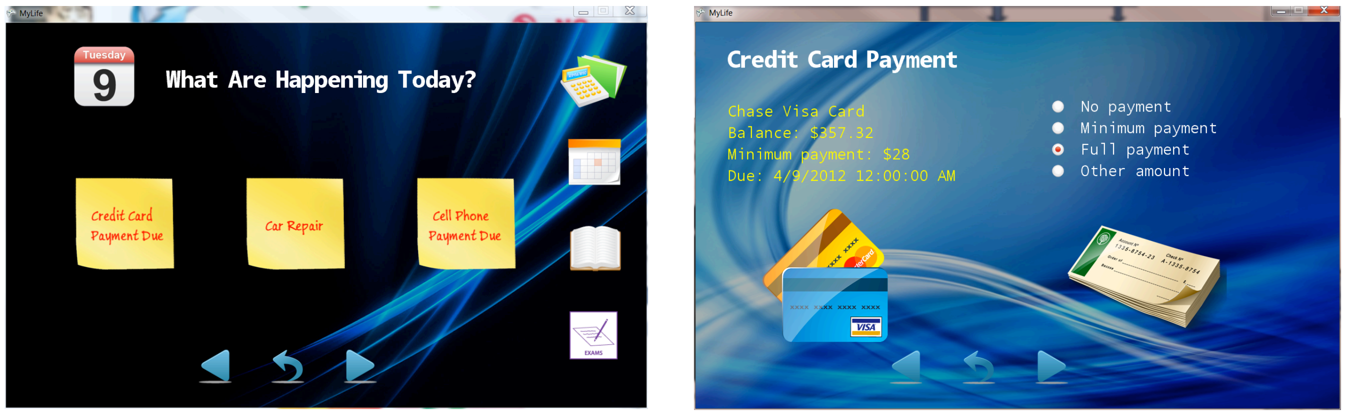


Figure 3. Architecture of game engine.

Currently, we are implementing three prototypes: (1) Web application using SilverLight as the front end, (2) XNA implementation to be deployed on Windows Phone and Xbox, and (3) Android implementation to be deployed on Android-based phones and tablets. Figure 4 shows screenshots of the current implementation using XNA. The game engine's event generator generates events based on user profiles at a configurable pace (e.g., a tick corresponding to a day, a week, or a month). Figure 4(a) shows the scenario where the user is presented with three events. When responding to these events, the user must make decisions, which will affect the user's personal financial situation in the game. Figure 4(b) shows that the user decides to pay credit card bill in full.



Credit Card Payment

Chase Visa Card
Balance: $357.32
Minimum payment: $28
Due: 4/9/2012 12:00:00 AM

No payment
Minimum payment
[✓ Full payment]
Other amount

(a) User is presented with events generated by game engine. (b) User responds to one of the events

Figure 4. Screenshots of game implementation using XNA.

We do not expect students to create the entire game with full features within a semester. Instead, we provide the game engine and several components to students and ask them to develop additional functions. Alternatively, students can create their standalone games or applications that address certain aspect of personal finance. For example students have created Android applications that estimate annual expenses for college, give quizzes about personal finance, and display online personal finance contents that are updated frequently through RSS feeds. Selected applications will be incorporated into the game by the research team.

EVALUATION

The implementation is evaluated systematically by an independent evaluator using a comprehensive methodology adapted from the AEIOU approach (Kemis and Walker 2000). AEIOU standards for Accountability, Effectiveness, Impact, Organization Context, Unanticipated Consequences. Data is collected using questionnaires, interviews, and observations as well as from sources such as enrollment and grades distribution. Surveys are being conducted both in the beginning and near the end of each semester. A preliminary analysis of the evaluation shows the approach has had some positive impacts on students' learning attitude and interest in computing. For example, in response to the question of "to which area(s) are you going to apply what you have learned from this course", most students expressed their readiness to apply what they had learned in "other related courses" or in "IT related jobs". When responding to the question of "are you interested in knowing more about game development and personal finance? Why", many students realized the importance of these areas. They understand that knowing game development is not only for fun in their learning process but also for an increase of their competiveness in their job opportunities.

The game will be used in FIN210 in Fall 2012. The effectiveness of game-based learning for students in the course will be evaluated using a similar method.

The preliminary findings of the ongoing evaluation study are consistent with the results from our prior experimental work in two courses. The teaching strategy has the potential to improve students learning outcomes in both computing and financial literacy.

CONCLUSION

We have designed, and are currently implementing, a novel teaching strategy to address challenges facing programming education. The approach integrates financial literacy into programming courses and adopts game-based learning in these courses. Building on the theory and empirical research on effective learning, the student-centered approach will significantly improve students' learning effectiveness by relating what the students learn to their lives. It will offer students hands-on experience of implementing financial literacy games that can be played online and on mobile devices.

ACKNOWLEDGEMENT

The work is supported in part by National Science Foundation under grant No. 1044845, ODU Office of Research, and ODU Center for Learning and Teaching. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation

REFERENCES

1. ACM, AIS, IEEE-CS. (2009) Curricula Recommendations, ACM.
2. American Association for the Advancement of Science. (2005) Invention and Impact: Building Excellence in Undergraduate Science, Technology, Engineering and Mathematics Education.
3. Bernard, T.S. (2009) In Grim Job Market, Student Loans Are a Costly Burden, *The New York Times*, New York, B6.
4. Cargenie Mellon University. (2009a) Alice, http://www.alice.org/index.php?page=alice_users/alice_users.
5. Cargenie Mellon University. (2009b) Educational Institutions Using Alice.
6. Damast, A. (1993) Asking for Student Loan Forgiveness, *BusinessWeek*, The McGraw-Hill Companies Inc.
7. Davis, B.G. (1993) Tools for Teaching, Jossey-Bass, San Francisco, CA.
8. Department of Education. (2009) National Postsecondary Student Aid Study, U.S. Department of Education.
9. Entertainment Software Association. (2009) Essential facts about the computer and video game industry: 2009 sales, demographic and usage data, Entertainment Software Association, Washington, DC.
10. Felder, R. and Peretti, S. (1998) A Learning Theory-Based Approach to the Undergraduate Laboratory, *ASEE Conference Proceedings*, Session 2413.
11. Kemis, M. and Walker, D. (2000) The a-e-I-o-u approach to program evaluation, *Journal of College Student Development*, 41, 1, 119-122.
12. NPD Group. (2010) 2009 U.S. Video Game Industry and PC Game Software Retail Sales Reach $20.2 Billion, http://www.npd.com/press/releases/press_100114.html.
13. Numedeon Inc. (2009) Whyville.
14. Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Devlin, M., and Paterson, J. (2007) A survey of literature on the teaching of introductory programming, *SIGCSE Bulletin*, 39, 4, 204-223.
15. Pearson Education Inc. (2009) Fun Brain.

16. Project Kaleidoscope. (2002) Recommendations for Action in Support of Undergraduate Science, Technology, Engineering, and Mathematics.
17. Ramírez, E. (2009) How the Recession Is Changing Students' College Plans, U.S. News & World Report.
18. Robins, A., Roundtree, J., and Roundtree, N. (2003) Learning and Teaching Programming: A Review and Discussion, *Computer Science Education*, 13, 2, 137-172.
19. Shen, Y. (2009a) Cross-Platform Game Development for Education and Training, *MODSIM World Conference & Expo*, Virginia Beach, VA.
20. Shen, Y. (2009b) Teaching Game Development Using Microsoft XNA Game Studio, *Spring Simulation Multiconference*, San Diego, CA.
21. Shen, Y. (2010) Introduction to Game Development, http://www.ece.odu.edu/~yshen/Game_2010.html.
22. Starrett, S., and Morcos, M. (2001) Hands-On, Minds-On Electric Power Education, *Journal of Engineering Education* 90, 1, 93-100.
23. The Learning Company. (2009) Reader Rabbit.
24. The National Academies. (2003) Improving Undergraduate Instruction in Science, Technology, Engineering and Mathematics.
25. Zhu, H. (2011) Teaching OOP with Financial Literacy, *IEEE Transactions on Education*, 54, 2, 328-331.