Middle-to-High School Girls as Game Designers – What are the Implications?

Magy Seif El-Nasr Ibrahim Yucel Joseph Zupko Andrea Tapia Brian Smith

College of Information Sciences & Technology

Penn State University
University Park, PA

{magy,iyucel,jzupko,atapia,bsmith}@ist.psu.edu

ABSTRACT

The percentage of young women choosing educational paths leading to science and technology-based employment has been dropping for several years. In our view, the core cause for this phenomenon is not a lack of ability, but rather a combination of low self efficacy, misconception of the IT field, and lack of interest and social support from families and peers. The specific aim of this paper is to discuss a case study – a class named *Gaming for Girls* . This class was offered to middle and high school girls three times from Fall 05 to Summer 06. In these classes, female students assumed the role of designers and developers engaged in developing their own games using commercial game engines. Based on this experience, we assert that through the activity of designing games using game engines, girls can (a) gain an understanding of the game development process, (b) acquire computer science skills, and (c) increase their confidence level with regards to computing.

Categories and Subject Descriptors

K.4 [**Computers and Education**], J.5 [**Arts and Humanities**]

General Terms

Performance, Design, Experimentation.

Keywords

Education, game modding, learning, gender

1. INTRODUCTION

The gender imbalance problem in the information technology (IT) and game industries has been a topic of interest for many years. In ′02 women accounted for 24.6% of the IT profession compared to 25.4% in ′96 [2] (not counting administrative jobs). A parallel can be found in the game industry where, in 2005, women accounted for only 11.5% of the game development workforce [3].

We regard this problem as a "pipeline" issue. Women earn significantly fewer undergraduate degrees in computer science and engineering than men. This may be traced back to middle and high school education, where women students continue to track out of math and science classes that provide the foundations for IT careers [4-6]. American cultural expectations and influences often convey the message that women are unsuitable for the IT world (e.g., [7]). In years prior to college, research studies have revealed effects of such social norms and expectations; for example, research showed that some girls exhibit low self-efficacy regarding computing and small amounts of informal and voluntary computer exploration (e.g., [8, 9]). It has been suggested that this is related to young women's negative perceptions of the IT field, e.g., it is male-oriented or anti-social [5].

As a result, there is a need for interventions that are aimed at increasing middle and high school girls' exposure to design and programming, thus demystifying the technology profession and promoting computer literacy. In this paper, we discuss a case study of such an intervention—*Gaming for Girls*—a game design course aim at engaging middle and high school students in design and programming activities through building games using existing game engines. Our premise is that this activity will (1) increase students' comfort level with technology, (2) demystify game and software development careers and underlying development processes, (3) increase students' self-efficacy with computing, and (4) promote the acquisition of programming and design skills.

In this paper, we will describe three different offerings of the *Gaming for Girls* course during Fall 05, Spring 06, and Summer 06. We present these offerings as case studies, detailing the curricula, engines used, problems encountered, and lessons learned. We collected surveys, interviews, and observational data during each offering. However, due to the small sample size (25 students in each offering) and the constant refinement of the curricula and game engines, it is difficult to provide generalized results, thus we will keep the analysis at an individual level. We will discuss learning outcomes based on our observations and interactions with students as well as their perception of what they learned based on survey data. In future work, beginning with the current offering (Fall 06), we will conduct and experiment with several assessment methods to measure learning outcomes.

2. Games and Learning

Many researchers have argued that games provide suitable environments for learning [10]. Several techniques have emerged from such studies: 1) learning through game playing, and 2) learning through game design, which has three flavours: creating games a) from scratch, b) using tools created by researchers, or c) using commercial game engine, i.e. game modding.

Modding is defined as the process of changing an existing game, thus it generally requires the use of a game with built-in development tools, e.g., *Unreal Tournament* and *Warcraft III*. Game modding demands an understanding of the underlying engine and game mechanics in order to use and modify the game, which is mostly learned through the use of the engine itself or by research on forums. Game modding has the advantage of offering content and mechanics, thus providing an architecture for creating complex and aesthetically pleasing games which are otherwise difficult to build given students' skill and time constraints.

Learning through design can occur in many domains with different types of development activities. Since the development of the Logo language in the 1960s, educational researchers have investigated ways that programming computers can facilitate learning about mathematics, computation, and more general problem solving skills [11-13]. Many researchers have devised approaches to engage students in learning through designing and developing their own games. For instance, Harel's work in elementary schools demonstrated children working for prolonged periods on the creation of educational games using the Logo programming language [14]. Kafai [15] noted similar engagement as students developed their own games, and she also tracked their abilities to incrementally create, evaluate, and revise their designs over time. Hooper's longitudinal study of software development in schools showed students expressing notions of cultural identity in their programs—ideas that were not likely to be expressed had students just played existing games [16].

These studies used Logo as the primary programming language, but a number of programming environments have been created to help novices learn by designing and implementing working computer programs [17-21]. The courses described in this paper are also examples of using games to teach computer science skills where students use commercial game engines instead of research-based languages or tools. Time, cost, and expertise are significant barriers to experimenting with video game design in educational settings, but customizing existing games may reduce the difficulty and make it possible for learners to create credible and aesthetically pleasing prototypes. The time commitment to return is important for middle and high school students since they generally lack the time to devote months to a game project but still desire 'commercial' aesthetics quality. In addition, using commercial game engines provides a robust infrastructure that students can use and a realistic environment that students can learn from (thus teaching them realities of game systems).

3. Gaming for Girls Courses

3.1 Engines used

A different game engine was used for each course offering. The choice of a game engine is critical, as it fundamentally promotes (or hinders) the course's learning objectives. As we previously argued, different engines promote different learning objectives [22]. Therefore, when choosing an engine, an educator needs to consider class schedule, size, style, student skills and age, in addition to the course's learning objectives. We chose three engines: *Warcraft III*, *Game Maker*, and *RPG Maker XP*.

Warcraft III was used in the Fall 05 course. *Warcraft III* includes a visual programming tool, the *Trigger Editor*, which allows students to program using dialogue boxes and point-and-click rather than writing code. However, it also allows students who are interested in writing code to do so through the same interface. Its programming environment includes notions of event-driven programming, Boolean logic, and parallel execution. Its art and design tools facilitate 2D map design, terrain design, and the creation of character behaviors. Additionally, it includes in-engine documentation in the form of tool tips and help text. These features may help students focus on semantics rather than syntax.

These features also had drawbacks, however. Semi-complex structures, such as deeply-nested conditionals, are tedious to specify in the visual programming tool. Additionally, the in-engine descriptions assume intermediate to advanced programming knowledge and make assumptions that may not be obvious, such as the fact that an expiration timer on floating text is dependant on the floating point "permanence" being off.

In the Spring 06 course, we used *Game Maker*, an engine that allows students to build 2D games. *Game Maker* is designed with the flexibility to build any type of game, and thus is not associated with a specific interaction model. Unlike *Warcraft III*, which embeds a real-time strategy interaction model, *Game Maker* can be used to produce a side-scroller as easily as a top-down role-playing game. While this greatly increases students' freedom and creativity, it can also be imposing as students needed to develop their own interaction model in addition to building a game. Using an existing game engine for game modding (*Warcraft III*) seemed to increase students' comfort with the tool when compared to creating a game from scratch as is necessary in *Game Maker*.

Game Maker offers a visual programming tool similar to *Warcraft III* with some differences. In *Game Maker*, programs are part of game objects but are also event driven. For example, a ball object can be programmed to reverse direction when a collision event occurs between the ball and a wall object. While the visual programming tool is simple to understand and use, it too becomes tedious to use with semi-complex structures.

Game Maker requires students to understand event-driven programming and a weak concept of object-oriented programming. Variables proved to be more important in *Game Maker* than in *Warcraft III* and parallel processing less important, although both concepts are present in both engines. Students must also understand geometry in 2D, sprites (pixel editing), and collision-detection, as *Game Maker* relies heavily on "object-collides-with-object" events.

In Summer 06, we used *RPG Maker XP*. Like, *Warcraft III* and *Game Maker*, it provides a visual programming tool on top of a scripting language (*Ruby* in this case). Code is event-driven, although the number and types of events is significantly smaller than in the other two engines. Although *RPG Maker XP* is not embedded in a game, it is defined by a 2D "Japanese-style" RPG interaction model, and thus is constrained by that model. This proved beneficial since students reacted favorably to the model, and thus it was easy for them to construct their games using this model as a base.

The visual programming tool of *RPG Maker XP* is conceptually different than its underlying scripting language. In fact, the tool is actually a "mini-language" that is implemented within *Ruby*. As a result, it was very difficult for students to move from the visual programming environment into *Ruby* when necessary, particularly when compared with *Warcraft III* or *Game Maker*.

Students working with *RPG Maker XP* deal with 2D map editing, layers (transparency), and event-driven programming. Switches, which are basically Boolean variables, are used extensively. Students will most likely need to deal with editing stats such as health and mana points to use the engine's combat system.

Table 1 summarizes the concepts that students are required to know in order to work with the engines.

Table 1. Programming concepts required for each engine

Game Engine	Programming Concepts Promoted
WarCraft III	Variables, Boolean logic, event-based programming, parallel execution, 2D map design, terrain design, and character behavior scripting
Game Maker	Variables, Boolean Logic, weak notion of Objects (as entities), sprites, collision detection, 2D geometry and coordinate systems
RPG Maker XP	Variables, Boolean logic, event-driven programming, concept of layers, 2D animation, 2D map design, and basic math for battle stats

3.2 Curriculum

The first offering used lectures to present knowledge and lab-time for developing games to deepen and solidify understanding. Later offerings nearly eliminated lectures all together, presenting knowledge through building mini-projects or other activities with the game engines. The last few days of all classes focused on providing students with an environment to finish and polish their game projects. Instructors concentrated on providing feedback, help, and facilitating discussion and critique.

The first course was offered over a 5-week period during Fall 05 using *Warcraft III*. The class met on Saturdays, once a week, for around 4 hours. Each week focused on a specific topic and students were given homework on that topic. The first week's topic covered map design. Students were asked to design and implement an environment for their games, motivated by a provided short story. The second week focused on characters and object design. Students were given a lecture on creating interesting characters in a narrative sense and asked to flesh their characters out on paper before implementing them in their game. Week three focused on character behavior and plot. This was their first exposure to programming, where they needed to make characters move, talk, and carry objects. The last two classes were spent providing students with more programming knowledge as needed and helping them debug. During the final class, students presented their games to parents and other educators.

The second course was offered over six weeks during Spring 06 using *Game Maker*. The class met on Saturdays, once a week, for 4 hours. The first class introduced *Game Maker* by asking students to build a simple game of *Breakout*, from start to finish. Students used existing art content, but many also created their own sprites for the project. During the second class, students focused specifically on designing environments and the collision of objects in environments. Week three introduced students to programming, and it was at this point that students decided whether they wanted to build a completely new game for the rest of the class, or to build on the *Breakout* game they had created during the first class. The last three weeks of the course were spent drawing and animating characters, polishing and critiquing, and providing students with programming concepts and debugging help as needed to complete their games.

The third course was offered as a 1-week camp during Summer 06 using *RPG Maker XP*. It should be noted that this particular offering engaged only middle school girls, while other offerings included both middle and high school girls. Similar to the second class, students built an entire game from scratch during the first day, this time a tale of King Arthur. Students were asked to bring a fable or myth to the class and spent the next four days telling that story in the game engine. The topics covered included map design and how to make interesting characters, as well as variables, flow control, and parallel execution.

4. Evaluation

We ran a study during each offering to evaluate the impact of the course on increasing self efficacy, engaging girls in design activities, promoting programming and design skills, and enhancing their perception of the IT field. In these studies, both quantitative and qualitative methods were used. Three types of data were collected: (1) surveys conducted at different time periods during the course sessions, (2) observations of student performance and questions during class periods, and (3) analysis of projects and assignments completed by the students.

Survey methods are often subjective, rely on perception, and to a large extent rely on the participant's judgement. However, they can be effective in measuring certain qualities, such as motivation and self-efficacy. The changes in the curriculum and engine prevent us from generalizing our findings.

The analysis presented in this section will be use the survey data taken from the summer 06 course. These surveys were comprised of both closed and open ended questions. In the case of the closed ended questions made of discreet categories, we present the data in numerical form, providing descriptive statistics. In the case of the open-ended questions, we provide the data in textual form, as illustrative quotes. These quotes are used both as stand alone qualitative data as well as supporting evidence for the numerical descriptive data, adding richness and meaning.

4.1 Capture and Motivate

On the first day of class, we asked students to talk about their motivations and hopes for the course. Most students expressed some excitement for creating a game. One student said, "After this morning's class, I'm excited to start working on more RPGs and perhaps even buy the program and make my own RPGs later." Other students expressed a desire to creatively bring their stories and characters to life. For example one student said she was most interested in, "…making my characters talk, building a world, and making an interesting story." When asked why they decided to take the course, they stated they liked computers (68%) and games (68%), and thought the class would be fun (61%). When asked how they felt about computers, 83% said they "loved them."

Parents were asked to complete a survey one week after the end of each course. When asked what long term effects the class had on their daughters, slightly more than half of the parents said they had noticed some change. A parent stated, "She learned the math she has been studying in school can have a real application. She learned programming can be fun."

It seems that game design motivated and captured the interest and attention of the students in our classes. The positive opinions from students must be tempered by the limitations of this study. This population was self-selected: students already had an interest in computers and gaming before they enrolled in the class or they would not have been interested. While the data says nothing about the effects this class might have on a truly general population, it obviously had some positive effect on this narrow, self-selected sample. This question demands further research.

4.2 Self Efficacy and Perception of IT

On the first day of class, the students were asked several questions to determine their confidence level with computers and their perceived self-efficacy. 24% felt they knew a lot about computers, 48% felt they knew [somewhat] a lot about computers. Fewer claimed they knew a lot about computer games. Fewer still felt they were confident with programming. Several expressed concern managing the programming aspects of the course. Another group of students expressed concerns being able to finish the project in the allotted time. One student said, "I don't know if I'll be able to finish a whole video game in 4 more days."

On the same day, we asked the students what they hoped to learn. The most common answer was to build video games. However, about a third of the students responded with the desire to learn more programming or computer skills. One student said she would like to learn, "… how to make an awesome video game. I want to learn everything about technology or at least more than I did." Another student stated that she simply wanted to learn, "how to be able to fix minor problems on my family's computer."

On the last day of class we asked the students similar questions about competency and self efficacy. 64% of students responded that they felt more confident about their abilities than they had on the first day, with 36% more stating they felt somewhat more confident. 96% felt they had learned a lot from the class. 48% felt they understood more about computer programming than on the first day with an additional 40% stating they felt somewhat more confident in their programming abilities. 52% felt they clearly understood how a computer game is built with an additional 48% giving more cautious assent. 60% felt very confident they could build a computer game in the future with an additional 24% feeling somewhat confident. Perhaps most importantly, 76% said they would like to take a more advanced programming class.

Before the course began, parents were surveyed on the impact of the *Gaming for Girls* class on their daughters. The majority of the parents hoped that their daughter would learn how to make a computer game (32%) or how to program a computer (28%). When the parents were asked what they imagined their daughter would be doing in the class, they unanimously answered learning how to create computer games using programming tools.

Parents were surveyed a second time one week after the end of each course. 88% of parents felt that the camp may have influenced their daughter's perception of working with computers, and confidence level with computers. When asked what long-term effects the class had on their daughters, slightly more than half of the parents said they had noticed some change. A mother stated that her daughter, "…has always been fairly comfortable with computers but she talks more about getting a Dell or converting one of our Macs with a PC emulator. The camp was clearly a confidence booster—something immeasurably important to girls of this age group." The parents also felt their daughters had gained technical skills. A mother of a student said, "she learned the basics of how games are made. She learned about various applications of computer technology and how computers are used in various areas."

Some parents have also expressed the impact of the course on their daughters' technology related activities and career choices. For example, a parent stated, "she was extremely enthusiastic about pursuing technology as a possible career choice. This is something that I will need to follow-up on to ensure that she is given the opportunity to explore. Additional classes would be of great interest." A mother of one of the students said about her daughter, "She wears her tee-shirt with confidence and talks often about her camp experience. She also talks more about enrolling in the College of [Information Sciences and Technology] and would like to explore possible scholarships, grants, and/or funding for that program." Another mother stated that her daughter, "… has purchased the software and is making new games already."

5. Discussion

Each offering resulted in many lessons that helped us reshape future offerings. These lessons were collected through student comments, discussions with individual students, observations, as well as the surveys and interview data collected.

The first two courses were offered during the school year (Fall and Spring). Our collected survey data indicated that girls were very busy and involved in many activities that competed with our course, including clubs, social activities, and of course, school. Spring was particularly busy, during which we had the lowest retention rate of the three classes. In general, many girls responded that they needed more time or would have liked to devote more time to their projects outside of class. Additionally, due to time commitment during the first two classes, it seemed that it was harder for students to assimilate the design and programming techniques. While all students demonstrated some understanding of the basics, such as Boolean logic, flow control, variables, and events as demonstrated by their project work, we felt that some left the class with holes in their knowledge or didn't fully understand some of the basics. On the other hand, some students demonstrated advanced knowledge beyond what was taught, such as using the scripting language in *Game Maker* to manipulate low-level parameters of game objects.

In the third class, all students understood most or all of the basic concepts we targeted, e.g., variables, Boolean logic, map design, mathematical manipulation to balance fight mechanics. Indeed, we entered this class severely underestimating their abilities and needed to add a great deal of material. For example, we presented a tutorial on creating sprites, including how to add highlights and shadow to sprites. Our initial plan did not include any discussion about creating sprites at all, which we added. Further, several students explored the *Ruby* programming language underlying the visual programming environment of *RPG Maker XP,* adding new features such as a visible timer. From analysis of projects and interactions with the students, we didn't note any student who intentionally avoided an idea or gave up on an idea because she found it too hard or lacked confidence that she would figure it out. Every game created had interesting game-play, map design, usage of music and sound effects, and a well told story. Student self-efficacy was very high.

The selection of the engine was also a very important choice. For example, students were constantly fighting *Warcraft III's* default interaction model as they tried to create their games. *Game Maker* posed the exact opposite problem. Students were presented with a blank slate, with no built-in interaction model to anchor their ideas and very little art content. *RPG Maker XP* seemed to strike the best balance, providing a great amount of content and a solid interaction model that was flexible enough to let students control their narratives.

In the summer course, we had time for polishing and critiquing, but we found that the girls had very little interest in revisiting their games. It seemed that the girls had little interest in reflecting on their games once they were completed.

Students were interested in drawing or otherwise creating their own characters throughout all of our classes (character modeling was a topic that came up unanimously as something the students would be interested in learning in the future). Sprite animation was covered in detail during the Summer class. However, once we showed them the steps involved, they generally lost interest, and very few students actually created their own characters. Instead, students tried to find the best fit among the provide content; for example, they would edit the sprites provided in terms of changing color hue, such changes involved much less time and effort than creating characters and sprites from scratch. A challenge for us in the future is to bring novel forms of visual control over characters into our classes that give our students the desired freedom without introducing a deterring time investment.

6. Future Work

The work presented here discussed three *Gaming for Girls* course offerings where we used a different engine for each offering. The courses provided a great environment for learning computer science skills. We have seen students apply basic programming concepts, such as variables, loops, and conditionals, and more advanced concepts, such as parallel and event programming as discussed in [23]. We can say with confidence that the majority of projects across all classes demonstrated the knowledge we targeted in the courses. However, to what degree students actually learned these concepts is unknown. This problem requires further research work. Future courses will include different assessment methods to gauge learned knowledge.

7. Conclusion

Over the three offerings of the *Gaming for Girls* course, data collected suggests that engaging girls in game design and development using commercial game engines can be used as a vehicle for (1) increasing students' self efficacy, (2) acquiring design, programming, and artistic skills, while (3) engaging them in the activity. However, more work is needed to generalize this assertion. We are continuing to run this class, and thus will continue to gather survey and observation data, which will help us generalize this assertion. In addition, we are currently exploring several assessment techniques to measure learning outcomes in future courses.

8. REFERENCES

[1] ITAA, "Adding Values: Growing Careers, ITAA's 2004 Workforce Study," Arlington, VA 2005.

[2] ITAA, "ITAA Blue Ribbon Panel on IT Diversity," 2003.

[3] IGDA, "Game Developers Demographic Report," 2005.

[4] T. Camp, "The Incredible Shrinking Pipeline," Communications of the ACM, vol. 40, 1997.

[5] J. Margolis and A. Fisher, "Greek Mythology and Attracting Undergraduate Women to Computer Science," presented at Joint National Conference of the Women in Engineering Program Advocates Network and the National Association of Minority Engineering Program Administrators, 1997.

[6] J. Margolis and A. Fisher, Unlocking the Clubhouse: Women in Computing. MA: MIT Press, 2002.

[7] E. M. Trauth, "Odd Girl Out: An Individual Differences Perspective on Women in the IT Profession," Information Technology and People, vol. 15, pp. 98-118, 2002.

[8] C. Beise, "IT Project Management and Virtual Teams," presented at Proceedings of the 2004 SIGMIS Conference on Computer Personal Research, Tucson, AZ, 2004.

[9] S. Nielsen, L. Von Hellens, and S. Wong, "The Game of Social Constructs: We're Going to WinIT," presented at Proceedings of the 2000 International Conference on Information Systems (ICIS), 2000.

[10] J. Gee, What Video Games Have to Teach Us About Learning and Literacy. NY: Palgrace Macmillan, 2004.

[11] S. Papert, Mindstorms: Children, Computers, and Powerful Ideas. New York: Basic Books, 1980.

[12] M. Resnick and S. Ocko, Lego/Logo: Learning through and about Design. Norwood, NJ: Ablex Publishing, 1993.

[13] R. D. Pea, D. M. Kurland, and J. Hawkins, "Logo and the Development of Thinking Skills," in Mirrors of Mind: Patterns of Experience in Educational Computing, R. D. Pea and K. Sheingold, Eds. Norwood, NJ: Ablex Pub., 1987.

[14] I. Harel, Children Designers. Norwood, NJ.: Albex, 1991.

[15] Y. Kafai, Minds in Play: Computer Game Design as a Context for Children's Learning. Mahwah, NJ: Erlbaum, 1994.

[16] P. K. Hooper, "They have their own Thoughts: Children's Learning of Computational Ideas from a Cultural Constructionist Perspective." Cambridge, MA: MIT, 1998.

[17] M. Conway, S. Audia, T. Burnette, D. Cosgrove, and K. Christiansen, "Alice: Lessons learned from building a 3D system for novices," presented at Proceedings of SIGCHI Conference on Human Factors in Computing Systems, 2000.

[18] A. Repenning and J. Ambach, "The Agentsheets Behavior Exchange: Supporting Social Behavior Processing," presented at CHI 97, New York, 1997.

[19] M. Resnick, Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds. Cambridge, MA: MIT Press, 1994.

[20] D. Ingalls, T. Kaehler, J. Maloney, S. Wallace, and A. Kay, "Back to the Future: The Story of Squeak, a Practical Smalltalk Written in Itself," presented at Proceedings of the 12th ACM SIGPLAN Conference on Object-oriented Programming Systems, Languages, and Applications, 1997.

[21] D. Smith, A. Cypher, and J. Spohrer, "Kidsim: Programming Agents without a Programming Language," Communications of the ACM, pp. 54-67, 1994.

[22] M. Seif El-Nasr and B. Smith, "Learning through Game Modding," presented at Games, Learning, and Society, Wisconsin, 2005.

[23] I. Yucel, J. Zupko, and M. Seif El-Nasr, "Education, IT, Girls, and Game Modding," International Journal of Interactive Technology and Smart Education, vol. 3, 2006.