A Design-Focused Analysis of Games Teaching Computer Science

Casper Harteveld, Northeastern University
Gillian Smith, Northeastern University
Gail Carmichael, Carleton University
Elisabeth Gee, Arizona State University
Carolee Stewart-Gardiner, Kean University

Introduction

In recent years, there has been a growing call to teach programming and computer science (CS) concepts more broadly than in university-level computer science programs. Initiatives such as Computer Science Education Week (Computer Science Education Week, 2014) and localized development training academies (Ada Developers Academy, 2014) aim to empower people from non-technical backgrounds to learn to code. Motivations for improving competency in CS range from training people for in-demand jobs, to diversifying the workforce, to realizing the benefits and broader applications of computational thinking. The Association for Computing Machinery (ACM) has called for a greater focus on CS education in K-12 (Tucker et al., 2006) aiming to demystify the field and encourage a larger and more diverse group of students to pursue computing as a career.

Many of these efforts to bring more people, especially younger students, into CS are aiming to address the "pipeline problem" that contributes to a dearth of women in CS. CS faces significant problems with diversity; despite women making up roughly 50% of computer and Internet users, they comprise only 27% of the tech industry workforce (United States Department of Commerce, 2011). Girls are given messages that computers are for boys (Margolis & Fisher, 2003) and are both subtly and actively discouraged from entering the field from an early age.

There have been many interventions to teach CS to young audiences, in both schools and informal learning contexts, with the aim to increase young girls' exposure to CS and foster shifts in attitude about the field. Many of these interventions have involved teaching students how to make games, using environments designed for younger audiences with no programming experience (Cooper, Dann, & Pausch, 2000a; Kelleher, Pausch, & Kiesler, 2007; Overmars, 1999; Resnick et al., 2009). Recent years, however, have also seen an increase in the number of games designed specifically to teach CS, as well as the use of games not designed for education as classroom activities.

This paper describes a preliminary study of how these CS educational games are designed in order to critically examine existing practices, given the importance of broadening participation in computing and especially improving gender diversity. With our research question "What are the common design features among games that teach computer science?" we have analyzed 36 games that have been used to teach CS, either in the classroom or in informal learning environments. We have identified 28 design features along which we compared the games. Several of these design features were chosen based on important features for gender-inclusive game design (Ray, 2004) as well as hypotheses for how women relate to CS, such as societal relevance (Denner & Campe, 2008; Denner, 2005).

As a result of the analysis, there are several common patterns that emerge from the games, ranging from the use of robots to the lack of collaboration. Our work has two main contributions to the literature on educational games for CS: 1) an analytical framework, grounded in theories of (educational) game design and CS education, within which any new game could be viewed, and 2) identification of a set of patterns in the design of educational games, which can be used in future work to evaluate the importance of design elements or to identify new opportunities for unique educational games.

Women and Gaming

There has been a great deal of research into the role of gender in games, in terms of issues in gender representation (Werner, Denner, Campe, & Kawamoto, 2012) and the games industry (International Game Developers Association, 2005), and how girls interact with games (Denner & Campe, 2008; Dickey, 2006; Fristoe, Denner, MacLaurin, Mateas, & Wardrip-Fruin, 2011; Kafai, Heeter, Denner, & Sun, 2008; Sykes, 2007). The notoriously poor representation of women and the use of violence as a common trope in mainstream games sustains the myth that women do not play games; however, 94% of teenage girls in the US play games compared to 99% of boys (Lenhart, Kahne, et al., 2008). Furthermore, 50% of adult women play games compared to 55% of adult men (Lenhart, Jones, & Macgill, 2008).

The concept of gender-inclusive game design is appealing due to its treatment of girls and boys as equals, rather than girls as a special, "niche" audience that requires special consideration and different kinds of games. Girls play every kind of game that exists, though there are cultural trends towards greater numbers of girls playing casual games, games addressing real-life issues, and games without a highly gendered or sexualized environment (Denner & Campe, 2008; Lenhart et al., 2008; Pugh et al., 2010).

Looking broadly at the literature, certain trends around what girls like and dislike in games begin to emerge (American Association of University Women, 2000; Gorriz & Medina, 2000; Hartmann & Klimmt, 2006; Schott & Horrell, 2000). One common theme is the enjoyment of puzzles, whether in support of a game's story or as a preferred genre in itself (Greenberg, Sherry, Lachlan, Lucas, & Holmstrom, 2010; Phan, Jardina, Hoyle, & Chaparro, 2012). Story is commonly believed to be important to girls, and this is indeed another element mentioned often. In some cases, story is directly mentioned as a desirable game element. More often girls want to see more instances of specific elements in games that may be best illustrated through story, such as humor, identification with or mimicking of principal characters, and meaningful dialog and character interaction. Games with some type of social element are popular, such as those that allow players to engage with social interactions (Heeter, Egidio, Mishra, Winn, & Winn, 2009). Unsurprisingly collaboration appears often as desirable, but interestingly, competition is not always considered a negative aspect of games. It seems that girls enjoy friendly competition (Jenson, de Castell, & Fisher, 2007). Many of these features of games were used when determining codes for the games analyzed in this paper, as described in the following section.

Methodology for Examining Games that Teach Computer Science

We have gathered a list of existing games--digital and analog--that teach CS. This list has been compiled from news articles, conferences attended by the authors, and word of mouth. Additional games were sought out through academic search engines such as Google Scholar and Scopus (using a combination of the keywords "Computer Science," "Teaching," Education," and "Game") and the game database MobyGames. This resulted in an initial list of 44 game initiatives to teach CS. Some of these games are explicitly designed as educational, while others are entertainment games that have been used in teaching CS (either formally or informally). Our next step was to determine whether those games were sufficient for consideration in our analysis, i.e. that they 1) are actually games; 2) have been used for teaching CS; and 3) are described in sufficient detail. In total 8 initiatives were excluded, resulting in a final list of 36 games (see Appendix). Choices were validated among the authors by cross checking the list.

The remaining games were more closely examined by coding them according to a list of codes developed based on our aforementioned literature review on women and games, CS education, and educational game design considerations (e.g., see Harteveld, 2011). The coding co-evolved while we examined the games, which is typical for most coding processes (Saldaña, 2012). Breaking the data apart in analytically relevant ways leads to further questions about the data, which then leads to generating categories, themes, and concepts, grasping meaning, and/or building theory. We performed three coding cycles, from rough to fine, with primary, secondary, and tertiary codes. The primary codes involve characteristics that are often used to describe games, such as the release date, the developers, availability, hardware platforms (if any), target audience, game genre(s), number of players, and camera perspective These primary codes do not reveal much about how the games are played, but do give the larger context about who has been developing these games, when, and for whom. The secondary codes are more extensive and are based on the literature on women and games and educational game design. Tertiary codes were used to characterize features of the game world, such as player interaction style, theme, and setting.

Gender-Inclusive Design

Based on the work on women and games we considered specifically if a *gender audience* was specified, and if so, if the game was targeted at males or females. Additionally, since inclusion of story or narrative seems of importance in how young women especially experience games (Denner & Campe, 2008; Ray, 2004), we coded how this was integrated into the game. For *story-based design*, we considered whether the game made use of no story, background story, cut-scenes, linear or non-linear storytelling.

Other gender-inspired codes concern the *problem-solving rhetoric* and the *problem context* that the game provides. For the rhetoric we considered if the game has an individual or collaborative orientation. In a game with an individual orientation, the player is the "superhero" who fixes the problem singlehandedly. In contrast, with a collaborative orientation the game rhetorically implies that CS is a collaborative field where the player's efforts combined with the efforts of others will help to solve the problem. The *problem context* specifies the relevance of solving the problem that the player is confronted with by the game. Some games offer none. Others state explicitly an individual relevance such as "programming is good for you!" Then there are games with a social relevance: solving the

problems will enable the player to help other people. Finally, there are games that have a broader societal relevance, such as in *ToonTalk 3*, where the player is tasked to help Marty the Martian who is trying to save the world.

Educational Aspects

From an educational game design perspective we tried to extract what the *learning objectives* of the games are, if any, and how this would help in teaching CS. We further coded the *educational intentions*. A number of games were originally created for entertainment and happened to be useful for teaching CS, such as *RoboRally*. Among the games purposefully developed to teach this, some state this explicitly to the players and others adopted a stealth learning approach. We define a stealth learning approach as one where the designers make no reference to what is being learned to the players. The learning happens without the learners realizing it. *Machineers* uses a stealth approach because the designers felt that otherwise children would be less eager to play the game (Lode, Franchi, & Frederiksen, 2013).

A closely related consideration is the specification (if any) of the *educational setting*. Some games are developed for specific curricula or the classroom in general (i.e., formal learning), whereas others are designed for outside the classroom (i.e., informal learning). Designing for either or both puts certain constraints on the design (Young et al., 2012), such as the *playtime*, which we coded in addition. Literature shows that games are used most effectively in combination with other educational material (Harteveld, 2012), so we considered the use of *accompanying material*. Some games stand alone and others come with a book or website (with either a variety of learning activities or just information).

As the rhetoric used in problem-solving may have consequences on the player's attitude and perspective on CS, so does the incorporation of a *problem-solving strategy*. An essential form of computational thinking is *algorithmic thinking* (Cooper, Dann, & Pausch, 2000b; Futschek, 2006). Algorithmic thinking involves thinking about problems generally, abstracting common traits so they can be treated as a class of problems instead of a single instance, and building sequences of instructions as solutions. From an educational point of view, such thinking can be fostered if more than one solution is possible. Otherwise players could treat the problems in isolation from each other. As single instances, each problem would have its own solution and no demands are made on the player to abstract the common traits for finding that solution. Providing a single problem-solving strategy does not mean that no algorithmic thinking skills are learned; we only hypothesize that it may be less likely. The last design consideration concerns what kind of *programming mechanic* is used (if any), which reflects a particular educational strategy and philosophy. Games can make use of:

- block code: predefined "blocks" of textual code that need to be mixed and matched;
- visual code: players manipulate objects to program;
- pseudo code: high-level, human readable code;
- unique code: code specifically developed for the game; or
- actual code: use of an existing programming language.

Design and Narrative

The tertiary codes were introduced during the coding process. The first involves *interaction style*. On close inspection it seemed relevant to specify how players interact with the game environment. What style has been implemented likely influences how players experience the game. We noted quiz, point-and-click, drag-and-drop, real-time control, and create-and-test interaction-styles. For analog games this category was not applicable (N/A). We became particularly interested in the design of the setting, theme, and protagonist. With regards to the *setting*, we coded if the game environment was abstract, realistic or iconic. This design choice is a standard consideration for visualizing the game world (Swink, 2009). We further noticed the recurrence of certain themes and started coding the *theme* of the setting: none, science fiction, fantasy, computer science-y, real-world metaphor (e.g., *C-Jump Programming* uses skiing as metaphor for how software programs work), or cartoon. We considered something "computer science-y" when it was not science fiction but did involve elements that are normally associated with this field, such as computers, technology, and robots. In fact, the use of robots was so prevalent throughout all the games that we decided to make it into its own category. The *robots?* code asked whether any robots were in the game. The final tertiary codes were about the protagonist. With the *protagonist* code we observed if the player manipulates the game without any role or context (None); assumes a role of some kind but does not control a

character such as being a scientist (Role-based); takes control of a character in a game (Character-based); or does not assume a role or take control of a character but is portrayed as something in the context of the theme of the game, such as being a snowboarder in *C-Jump Programming* (Metaphor-based). We introduced the last code, *protagonist content*, to consider what type of protagonist the player controls (if any). The codes for this category are: abstract, robot (or cyborg), animal, male, female, non-gender specific, and choice.

Preliminary Results

With our methodology we were able to distill preliminary patterns that allow for reflection on how the games are designed. We will discuss ten of these patterns, in no particular order of importance:

1. *Robots rule:* The inclusion of robots was so pervasive that we ended up making it its own code. Robots are easily associated with the field of computer science, given the prevalence of robots in popular media and science fiction. While the design and programming of robots does involve some CS skills, in actual practice the development of robots is more the domain of mechanical, computer and electrical engineering, and encompasses only a small fraction of what computer scientists are preoccupied with.

2. *Puzzle instinct:* Almost all games could be associated with the puzzle game genre. Some were straight up puzzle games, providing the player a single screen with a puzzle to solve. Others combined the puzzle genre with that of the adventure genre, allowing the player to walk around and then solve puzzles occasionally. Puzzles lend themselves well to CS, as much of this discipline is about solving complex problems logically and rationally. Also, puzzle games are in the top two genres of games played by teens (Lenhart et al., 2008), possibly because humans have an innate "puzzle instinct" (Danesi, 2002). This puzzle instinct appears to exist for choosing a game genre for designing CS educational games too.

3. *The single hero:* Except for *Bots*, which is described as a social collaboration-game where players work together to solve puzzles using simple programming concepts, all *digital* games we included are single-player games. The analog games were all competitive multiplayer games. This choice for a single-player game may have consequences on how players view CS. One of the "myths" about computer scientists is that the work is solitary and boring (Drexel University Department of Computer Science, n.d.). In practice, computer scientists must work together and they (should) talk with users often (Nielsen, 1993). To be fair to developers, many puzzle games are single-player, so if an early design choice is to use the puzzle genre, it may automatically result in the game being a single-player game. In addition, multiplayer games are much harder to develop (Harteveld & Bekebrede, 2011). However, this design decision may have implications on problem-solving rhetoric: across all games, we observed that problem-solving was solely individually oriented. This rhetoric further undermines the idea that CS is about collaboration.

4. *Born late:* Games teaching CS are largely a phenomenon from the past decade (and especially the past five years), whereas educational programming languages have been around much longer. For example, *Logo* was designed in 1967 (Logo Foundation, n.d.). The development of games to teach CS has even lagged behind the design of educational games in general (Harteveld, 2011). This relatively recent interest may be due in part to the continued emphasis on using game design (not game play) to teach CS skills, primarily programming.

5. *University-led development:* Researchers and/or students have created almost all of the included games. Although this pattern reflects much of the development of educational games in the past years, for-profit companies have developed games about mathematics and language, for example. The difference possibly reflects the place and status of CS in society. The viewpoint that programming and computational thinking are necessary skills for the 21st century is increasingly popular but not yet reflected in most school curricula. Since for-profit companies' priorities tend to be market-driven, a more widespread interest in the design of games to teach CS may not develop until computational thinking or CS concepts are more widely incorporated into K-12 and higher education.

6. *Broad and unspecific learning objectives:* The learning objectives of most games amount to "teaching programming" or the "basics of programming". There are many concepts involved in learning how to program and, therefore, such learning objectives can be considered broad and unspecific. This is especially problematic for assessing student learning outcomes—either by the game or by a teacher. It might be that within the game's design, these broad objectives have been decomposed into more detailed objectives, which (though implicit to the player) then have guided game design and, potentially, assessment. But it could also be, and this is not an unlikely scenario, that this lack of clarity is reflected in the game's design, and then it becomes questionable if the game is effective at all. Games that have specific, detailed learning objectives typically are highly specialized and teach a particular CS concept. This is illustrated by *Not! The Simpsons Donut Drop* that teaches basic Boolean expressions only.

7. *Gender ambiguity:* Many games specify the target audience in terms of age, not in terms of gender. One of the tropes in commercial games is a strong emphasis on male characters, but if we look at the protagonists in the games we reviewed, we see a preference for non-gender specific characters. The one game with a female protagonist was led by a female game designer—but even for this game the intended gender audience was not specified.

8. *Lack of context:* Very few of the games studied feature any kind of story, even a background story. Though puzzle games do not need story, it can provide meaning and relevance to any actions the player makes. This may be important for teaching CS and building a favorable impression of the field, as social context and relevance is important for learners, especially women (Margolis & Fisher, 2003). The games with at the very least a background story often provide a problem with either societal relevance (e.g., *ToonTalk 3*) or social relevance (e.g., *Machineers)*.

9. *Dominance of fiction*: Amongst all games, the use of fiction of any kind is predominant. Players set foot in a science fiction, fantasy, or cartoon setting and control an avatar that represents a character or something along the lines of the metaphor (e.g., turtle or snowboarder). This dominant use of fiction is not necessarily what all educational games turn to. Some realism may be incorporated by letting the player take the role of a scientist or a professional, or the setting may be depicted in a realistic yet iconic style. CS involves actions in a digital, abstract context (i.e., programming) and thus might be more challenging to represent in a compelling, real world context than, say, political science or biology. However, finding ways to contextualize CS in real-world contexts may be important to help students envision CS as a potential career.

10. *Proliferation of coding:* We have not found a strong pattern of programming mechanics. Of those games that include the ability to program, practically any variety of block, visual, pseudo, unique, or actual coding can be found. This is likely a result of different philosophies on teaching CS, the games having different target audiences, and some games being geared to teach a specific language. Nevertheless, this proliferation begs the question of what programming mechanic is most appropriate for particular learning objectives, and what mechanic designers should choose going forward.

Conclusion and Future Work

Our critical analysis provides grounded reasons to question how games teaching CS have been designed, especially in terms of gender-inclusivity. It also allows for identifying new opportunities for unique educational games. We identified opportunities to emphasize collaboration, include narrative, and provide a (social/societal) relevant and less fantastical context, all factors in designing a game that appeals to girls. Our work additionally encourages designers to become more specific about the underlying educational philosophy and associated objectives, and be more creative with the game genre and setting. CS is a far broader field than can be effectively portrayed with games that are just about individuals solving puzzles and programming robots, and while many educational games are including options for players to interact with a diverse cast of characters, there is a great need for diversity in the actual *mechanics* of the games.

Our methodology and accompanying results provide game designers and educators an incentive to adopt a critically reflective and creative attitude toward teaching CS through games. The methodology can be used as a checks-and-balances system during design, much akin to Flanagan's (Flanagan, 2009) critical play model that encourages the subversion of popular gaming tropes through new styles of game making. The difference here is that we encourage subversion of popular *educational* gaming practices. Our outcomes are specific to CS educational games, but we believe that our approach and insights can be extended to educational games on other topics and for considering other issues of inclusivity, such as cultural differences (Hofstede, 1984).

The work presented here is preliminary. We intend to extend and fine-tune our methodology with more categories and codes, consider more games, and use these patterns as part of future research into understanding the efficacy and design of CS educational games. Our team's long-term goal is to create a gender-inclusive game for an international audience of middle school-aged students (especially girls) that teaches a wide variety of CS concepts rather than simply teaching programming, and showcases the variety of ways in which CS can be applied to show the broader relevance of the field.

References

Ada Developers Academy. (2014). Retrieved from http://adadevelopersacademy.org/

American Association of University Women. (2000). *Tech-savvy: Educating girls in the new computer age*. Ameri can Association of University Women.

Computer Science Education Week. (2014). Retrieved from http://csedweek.org/

Cooper, S., Dann, W., & Pausch, R. (2000a). Alice: a 3-D tool for introductory programming concepts. *Journal of Computing Sciences in Colleges*, *15*(5), 107–116.

Cooper, S., Dann, W., & Pausch, R. (2000b). Developing algorithmic thinking with Alice. In *The Proceedings of ISECON 2000* (Vol. 17, pp. 506–539).

Danesi, M. (2002). *The puzzle instinct: the meaning of puzzles in human life*. Bloomington, IN: Indiana University Press.

Denner, J. (2005). Girls creating games: Challenging existing assumptions about game content. Presented at the DiGRA: Changing Views: Worlds in Play, Vancouver, BC, Canada.

Denner, J., & Campe, S. (2008). What games made by girls can tell us. In *Beyond Barbie and Mortal Kombat: New perspectives on gender and computer games*. Cambridge, MA: MIT Press.

Dickey, M. D. (2006). Girl gamers: the controversy of girl games and the relevance of female-oriented game design for instructional design. *British Journal of Educational Technology*, *37*(5), 785–793.

Drexel University Department of Computer Science. (n.d.). Myths and Facts about Computer Science. Retrieved from https://www.cs.drexel.edu/undergraduate/cs/about/faq

Flanagan, M. (2009). *Critical play: radical game design*. Cambridge, MA: The MIT Press.

Fristoe, T., Denner, J., MacLaurin, M., Mateas, M., & Wardrip-Fruin, N. (2011). *Say it with systems: expanding Kodu's expressive power through gender-inclusive mechanics*. Bordeaux, France: ACM. Retrieved from http://doi.acm.org/10.1145/2159365.2159396

Futschek, G. (2006). Algorithmic Thinking: The Key for Understanding Computer Science. In R. T. Mittermeir (Ed.), *Informatics Education: The Bridge between Using and Understanding Computers* (pp. 159–168). Springer Berlin Heidelberg.

Gorriz, C. M., & Medina, C. (2000). Engaging girls with computers through software games. *Communications of the ACM*, *43*(1), 42–49.

Greenberg, B. S., Sherry, J., Lachlan, K., Lucas, K., & Holmstrom, A. (2010). Orientations to video games among gender and age groups. *Simulation & Gaming*, *41*(2), 238–259.

Harteveld, C. (2011). *Triadic game design: Balancing reality, meaning and play*. London, UK: Springer.

Harteveld, C. (2012). *Making sense of virtual risks: A quasi-experimental investigation into game-based training*. Amsterdam, the Netherlands: IOS Press.

Harteveld, C., & Bekebrede, G. (2011). Learning in Single-Versus Multiplayer Games: The More the Merrier? *Simulation & Gaming*, *42*(1), 43–63.

Hartmann, T., & Klimmt, C. (2006). Gender and computer games: Exploring females' dislikes. *Journal of Comput er-Mediated Communication*, *11*(4), 910–931.

Heeter, C., Egidio, R., Mishra, P., Winn, B., & Winn, J. (2009). Alien Games Do Girls Prefer Games Designed by Girls? *Games and Culture*, *4*(1), 74–100.

Hofstede, G. (1984). *Culture's consequences: International differences in work-related values* (Vol. 5). Thousand Oaks, CA: Sage.

International Game Developers Association. (2005). *Game Developer Demographics: AN Exploration of Work*

place Diversity. IGDA

Jenson, J., de Castell, S., & Fisher, S. (2007). Girls playing games: rethinking stereotypes. In *Proceedings of the 2007 conference on Future Play* (pp. 9–16). ACM.

Kafai, Y. B., Heeter, C., Denner, J., & Sun, J. Y. (2008). Pink, purple, casual, or mainstream games: moving beyond the gender divide. In *Beyond Barbie and Mortal Combat: New Perspectives on Gender and Gam ing* (pp. xi–xxv). Cambridge, MA: The MIT Press.

Kelleher, C., Pausch, R., & Kiesler, S. (2007). Storytelling alice motivates middle school girls to learn computer programming. In *Proceedings of the SIGCHI conference on Human factors in computing sys tems* (pp. 1455–1464). San Jose, CA.

Lenhart, A., Jones, S., & Macgill, A. (2008). *Adults and Video Games*. PEW Internet and American Life Project. Retrieved from http://www.pewinternet.org/Reports/2008/Adults-and-Video-Games.aspx

Lenhart, A., Kahne, J., Middaugh, E., Macgill, A. R., Evans, C., & Vitak, J. (2008). *Teens, Video Games, and Civics*. Pew Internet & American Life Project.

Lode, H., Franchi, G. E., & Frederiksen, N. G. (2013). Machineers: Playfully Introducing Programming to Children. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems* (pp. 2639–2642). New York, NY, USA: ACM. doi:10.1145/2468356.2479483

Logo Foundation. (n.d.). *What is Logo? The Logo Programming Language*. MIT Media Lab. Retrieved from http://el.media.mit.edu/logo-foundation/logo/programming.html

Margolis, J., & Fisher, A. (2003). *Unlocking the Clubhouse: Women in Computing*. MIT Press.

Nielsen, J. (1993). *Usability Engineering*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Overmars, M. (1999). *GameMaker*. YoYo Games.

Phan, M. H., Jardina, J. R., Hoyle, S., & Chaparro, B. S. (2012). Examining the Role of Gender in Video Game Usage, Preference, and Behavior. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (Vol. 56, pp. 1496–1500). SAGE Publications.

Pugh, K. J., Linnenbrink-Garcia, L., Koskey, K. L. K., Stewart, V. C., & Manzey, C. (2010). Motivation, learning, and transformative experience: A study of deep engagement in science. *Science Education*, *94*(1), 1–28. doi:10.1002/sce.20344

Ray, S. G. (2004). *Gender Inclusive Game Design: Expanding the Market*. Cengage Learning.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., … Kafai, Y. (2009). Scratch: programming for all. *Commun. ACM*, *52*(11), 60–67.

Saldaña, J. (2012). *The coding manual for qualitative researchers*. Thousand Oaks, CA: Sage.

Schott, G. R., & Horrell, K. R. (2000). Girl gamers and their relationship with the gaming culture. *Convergence: The International Journal of Research into New Media Technologies*, *6*(4), 36–53.

Swink, S. (2009). *Game feel: a game designer's guide to virtual sensation*. Burlington, MA: Morgan Kaufmann.

Sykes, E. R. (2007). Determining the effectiveness of the 3D Alice programming environment at the Computer Science I level. *Journal of Educational Computing Research*, *36*(2), 223–244.

Tucker, A., Deek, F., Jones, J., McCowan, D., Stephenson, C., & Verno, A. (2006). *ACM K-12 CS Model Curricu lum, 2nd Edition*. Computer Science Teachers Association.

United States Department of Commerce. (2011). *Women in STEM: A Gender Gap to Innovation*. Retrieved from http://www.esa.doc.gov/Reports/women-stem-gender-gap-innovation

Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012). The fairy performance assessment: measuring computational thinking in middle school. In *Proceedings of the 43rd ACM technical symposium on Com puter Science Education* (pp. 215–220).

Young, M. F., Slota, S., Cutter, A. B., Jalette, G., Mullin, G., Lai, B., … Yukhymenko, M. (2012). Our Princess Is in Another Castle: A Review of Trends in Serious Gaming for Education. *Review of Educational Research*, *82*(1), 61–89. doi:10.3102/0034654312436980

Appendix: Games Used in Analysis

1. **(untitled):** Moradi, J. (2013). A Card Game to Teach Kids Computer Science and Math. http://javaunmoradi.com/blog/2013/01/22/a-card-game-to-teach-kids-computer-science-and-math/.
2. **A.I. Wars: The Insect Mind:** Tactical Neuronics (2012). *A.I. Wars: The Insect Mind (PC)*. http://www.tactical-neuronics.com/content/main.asp.
3. **Beadloom:** Boyce, A., & Barnes, T. (2010). BeadLoom Game: Using game elements to Increase motivation and learning. In *Proceedings of FDG*, 25–31. New York, NY, USA: ACM.
4. **Blockly: Maze:** Google Blockly Project (n.d.). *Blockly: Maze (Web)*. BrainPOP. http://www.brainpop.com/games/blocklymaze/.
5. **BotLogic:** BotLogic.us (2013). *BotLogic (Web)*. http://botlogic.us/.
6. **BOTS:** Hicks, D., Catete, V. Culler, D., & Kingsley, N. (n.d.). *BOTS (PC Game)*. Game2Learn http://www.game2learn.com/?page_id=63.
7. **C-Jump:** C-Jump Factory (2007). *C-Jump*. http://www.c-jump.com
8. **Cargo-Bot:** Two Lives Left (2012). *Cargo-Bot (iOS)*. http://twolivesleft.com/CargoBot/.
9. **CeeBot:** Epsitec Games (n.d.). *CeeBot (PC)*. http://www.ceebot.com/ceebot/index-e.php.
10. **CodeSpells:** Esper, S., Foster, S.R., & Griswold, W.G. (2013). CodeSpells: Embodying the metaphor of wizardry for programming. In *Proceedings of ITiCSE '13*, 249–254.New York, NY, USA: ACM,
11. **CoLoBot:** Epsitec Games (n.d.). *CoLoBot (PC).* http://www.ceebot.com/colobot/index-e.php.
12. **Dance Tool:** Campbell, A., Culler, D., Boyce, A., Hicks, D., Sabo, D., Powell, E., & Pickford, S. (n.d.). *Dance Tool (PC Game)*. Game2Learn, http://www.game2learn.com/?page_id=108.
13. **GameStar Mechanic:** Institute of Play, & E-Line Media (2010). *Gamestar Mechanic (Web)*. https://gamestar-mechanic.com/.
14. **LearnMem1:** Papastergiou, M. (2009). Digital game-based learning in high school computer science education: Impact on educational effectiveness and student motivation. *Computers & Education*, 52(1), 1–12.
15. **LightBot:** LightBot Inc. (n.d.), *LightBot (iOS & Android)*. http://light-bot.com
16. **Logical Journey of the Zoombinis:** Broderbund Software (2001). *Logical Journey of the Zoombinis (PC Game)* (version 2.0). The Learning Company,
17. **Machineers:** Frederiksen, N. G., Lode, H., Giuseppe Enrico Franchi, G.E., & Fischerson, M. (2012) *Machineers (PC Game)*. Copenhagen, Denmark: Lohika Games..
18. **Manufactoria:** PleasingFungus Games (2011). *Manufactoria (Web)*. http://pleasingfungus.com/.
19. **MindRover: The Europa Project:** CogniToy (2000).*MindRover: The Europa Project (PC)*. CogniToy.
20. **Move the Turtle:** Next is Great (2012). *Move the Turtle (iOS)*,. http://movetheturtle.com/.
21. **Not! The Simpsons Donut Drop:** Edgington, J.M. (2010). Toward using games to teach fundamental computer science concepts. PhD Thesis, University of Denver.
22. **Recursive Breakout:** (same ref as #21 above)
23. **Ricochet Robots:** Randolph, A. (1999). *Ricochet Robots (Board Game)*. Rio Grande Games.

24. **RoboRally:** Garfield, R., & Wizards of the Coast (1994). *RoboRally (Board Game)*,

25. **Robot Odyssey:** Wallace, M., & Grimm, L. (1984). *Robot Odyssey (Apple II)*. Learning Company.

26. **Robot Turtles:** Shapiro, D. (2013). *Robot Turtles*. . http://www.robotturtles.com/.

27. **RoboZZle:** Ostrovsky, I. (n.d.). *Robozzle (Web)*, http://www.robozzle.com/.

28. **Rocky's Boots:** Robinett, W., & Grimm, L (1982). *Rocky's Boots (Apple II)*. Learning Company.

29. **Super Solvers: Gizmos and Gadgets:** Learning Company (1993). *Super Solvers (PC)*.

30. **The Lost Mind of Dr. Brain:** Sierra Entertainment (1994). *The Lost Mind of Dr. Brain (PC)*,.

31. **ToonTalk 3:** Animated Programs (n.d.). *ToonTalk*. Http://www.toontalk.com/English/toontalk.htm

32. **Totally Spies! Swamp Monster Blues:** FiniteMonkey Inc. (2007). *Totally Spies! Swamp Monster Blues (PC)*. Brighter Minds Media Inc.

33. **Tynker: Lost in Space:** Tynker (n.d.). *Tynker: Lost in Space, (Web)*. BrainPOP. http://www.brainpop.com/games/tynkerlostinspace/.

34. **Tynker: Puppy Adventure:** Tynker (n.d.). *Tynker: Puppy Adventure (Web)*. BrainPOP. http://www.brainpop.com/games/tynkerpuppyadventure/.

35. **Tynker: Sketch Racer:** Tynker (n.d.). *Tynker: Sketch Racer (Web)*. BrainPOP. http://www.brainpop.com/games/tynkersketchracer/.

36. **Wu's Castle:** Eagle, M., & Barnes. T. (2008). Wu's Castle: Teaching arrays and loops in a game. In *Proceedings of ITiCSE '08*, 245–249. New York, NY, USA: ACM