Need a tittle

****

Anonymized Author(s)

ABSTRACT

Need to re-write.

CCS CONCEPTS

• **Applied computing** → **Interactive learning environments**; • **Social and professional topics** → *Computational thinking*; *CS1*; *Computing literacy*.

KEYWORDS

Programming language education; Cybersecurity education; Web application; Card game; Game-based Learning

ACM Reference Format:
Anonymized Author(s). 2020. Need a tittle: ************************. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Needs to update.

Most games and environments developed with the intention of teaching computer programming concepts do so by requiring the student to either learn an existing programming language (e.g. Java [11], Python [3, 6] or JavaScript [3, 5, 7]) or a language specific to the learning environment (e.g. Scratch [31] or Alice [30]). If the goal is to teach the fundamental concepts of computer programming, this requirement can result in a situation where the learner feels intimidated, confused or frustrated when their "program" does not work correctly. Also, as Bromwich et al. noted, many of these educational programming languages and environments are "often ambitious in what they are trying to teach beginning programmers. They even go as far as to try teaching concurrency, something with which even advanced programmers often have difficulty." [13].

Also, such learning games and environments commonly use either a puzzle-solving premise, such as navigating a maze, or a sandbox paradigm, where a learner can utilize the language in an open-ended manner. As puzzle-solving activities tend to favour a specific demographic [23], and the sandbox paradigm requires oversight by an outside influence (i.e. an instructor) to ensure learning progression [13], both approaches have significant drawbacks.

"Game Name"[1] was created to address these concerns by providing a web-based card game that teaches or reinforces the fundamental concepts of programming and cybersecurity to those with limited or no programming or computer security experience [17]. Players use various cards to create a program that executes a target number of instructions while launching cyberattacks against opponents and using representations of cybersecurity tools to defend themselves. Players are awarded additional points according to how they construct their program.

Several limitations and areas for improvement were identified for the initial version of "Game Name" following both formal [17] and informal observations and discussions with players. The specific limitations identified included: confusion regarding the concept of method/function/procedure, the overgeneralization of cyberattacks, and frustration due to the randomness of the "conditional statement" game mechanic.

This paper presents a redesign of "Game Name" which addresses these limitations, resulting in "Game Name" *v.*2.0, a version which we believe better communicates the principles of functional decomposition, conditional statements, and real-world cybersecurity concepts. Also, "Game Name" *v.*2.0 introduces learners to two key classes of algorithms: searching and sorting.

The paper proceeds by presenting a survey of card or board games whose intent is similar to that of "Game Name" , followed by a description of the gameplay and cards found in "Game Name" *v.*2.0.

2 RELATED WORK

There are several research works have been conducted regarding the use of games for teaching computer programming and software engineering.

2.1 Game-Based Learning research in SE

The following is an overview of the previous work regarding Game-Based Learning in the field of software engineering.

Tao et al. [28] emphasize learning software engineering through different gaming approaches. In their research, they found that gaming for software engineering education required a slightly different approach than the traditional Game-Based Learning methodologies. They created Pex4Fun to serve both the social aspects of Game-Based Learning and the presentation of software engineering content. The learning outcome from the research work is gaming and entertainment can be a source of education and that interactive learning has excellent value. Another discovery is that learning while playing can be effective in the industrial field.

Swapneel et al. [26] discussed how highly addictive socially optimized (HALO) provides concentration through an adaptive environment. The game environment is developed in such a way so that the player can enjoy the work in a gaming atmosphere. The initial stage is known as 'Quest', which is a preliminary introduction

Conference'17, July 2017, Washington, DC, USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

[1] Project repository URL removed for blind-review.

of the system. Here someone senior must work voluntarily or can be assigned. If the quest is harder for a single player, it could be done in a team, and then it is called a party. Another essential aspect is context switching. If an employer works on different modules then it will require more time, but HALO groups the same type of coding all together so that less context switching is required. Both the academic and industrial fields can be beneficial by adopting the game-based approach in software engineering. Evaluation of the technique was not present, which is one of the drawbacks of this research.

Miljanovic et al. [21] discussed Robobug, a debugging technique through gaming in their research paper. Debugging is an essential tool in programming. Many good programmers struggle to find the bugs in a code segment. Debugging requires practice and patience; both might be difficult for a new programmer. Many computer science students struggle with debugging and feel left behind. Robobug presents different debugging techniques at different levels.

Szabo [27] proposed GameDevTycoon for teaching software engineering. Based on their work, there are thirteen different criteria for Game-Based Learning. GameDevTycoon covers most of the criteria. Gameplay analysis and software process models are simultaneously covered in the game. Three primary stages of software development are taught through the gameplay. The initial stage is known as the garage, the second stage is called team management, and the final stage is known as world domination. Each stage has separate responsibilities; for example, in the initial stage in software development, one should focus on the quality and latest research.

Pieper et al. [25] presented a case study of the Software Engineering Method and Theory (SEMAT) to identify the educational outcomes in Digital Game-Based Learning. SEMAT is a part of the emerging OMG[2] standard [18]. The case study shows that the evaluation of a developed integrated scenario can provide an in-depth analysis of the result. Nevertheless, the data was not sufficient to reach a conclusive decision regarding the pattern of learning. As a result, SEMAT was not referred to as a standard.

Mauricio et al. [16] identified a different methodology that can be or is already applied in different interactive games for Software Engineering (SE) Game-Based Learning. Also, they explored different primary studies related to SE education, found the learning outcomes and mapped those outcomes to different SE project stages.

Vladimir et al. [32] discussed the different classifications of gameplay and their impact on several learning criteria. According to the researchers, gamification is a growing area in the business industry. The researchers emphasized the SWOT (Strengths, Weaknesses, Opportunities, Threats) framework [24] to find the learning criteria. However, they found that creating a software engineering Game-Based engine is a challenging task, one which programmers and industry should give a high priority.

3 "GAME NAME" *V.*3.0 GAMEPLAY ADDITIONS

This section describes the changes and additions *Agile* mode brings to "Game Name" *v.*3.0 . These additions break the game into four phases requirements, planning, implementation, and testing. Each phase is designed to introduce some of the concepts of the *Agile* process and the software development lifecycle. Players still compete for points, but the game is broken into three 10 turn rounds called sprints. The game automatically ends when all three sprints have been completed. Players still build programs to get points, but the standard mode objectives are replaced by individual requirements with one objective for each sprint. Completing these objectives awards additional points to a player that are used in addition to their instruction score to determine a winner. Each player now chooses their own set of requirements complete and builds their own customizable deck to help them complete those requirements. Each of the additional elements and any UI additions or changes will be described in the appropriate phase below.

3.1 Game Setup

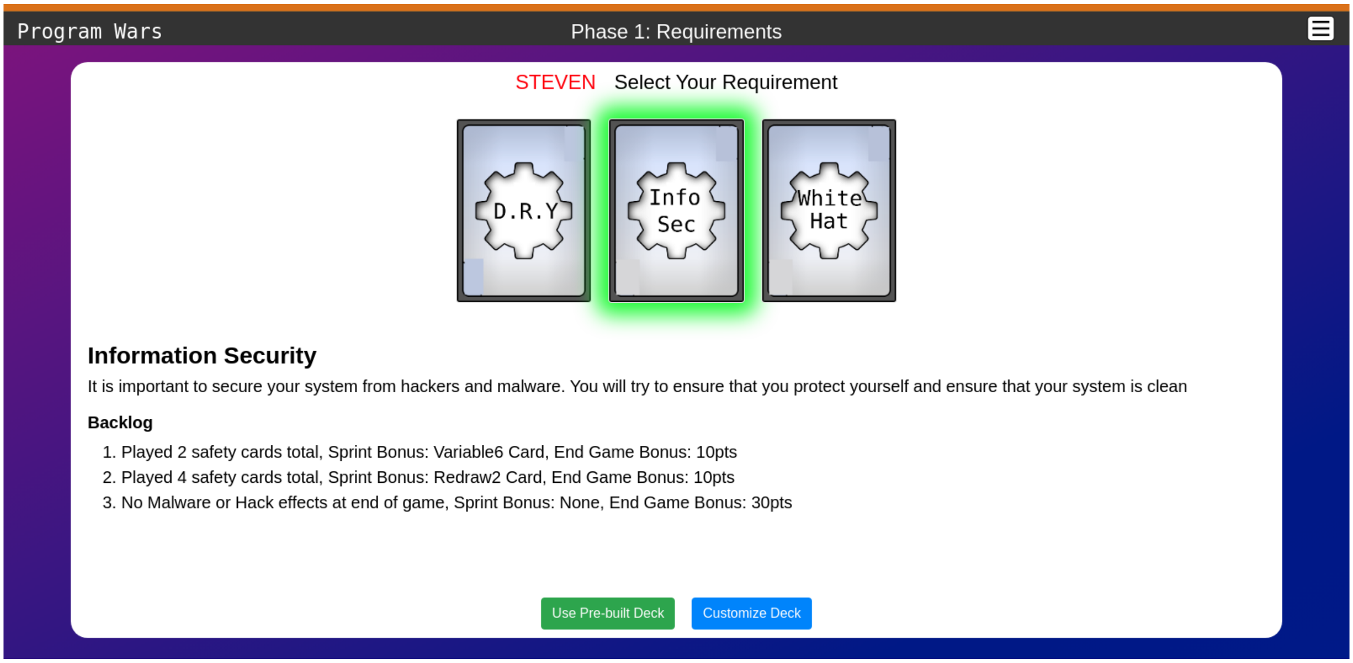
In "Game Name" *v.*3.0 the mode dropdown will have an additional entry in it for *Agile* mode. When selected it will give a short description to let players know their goals will change to completing individual requirements. When this mode is selected the level dropdown will be disabled as players will use individual decks with cards related to the requirement they will choose. However, it may in the future be used to select AI difficulty levels for the *Agile* mode. As in other modes when an acceptable number of players has been added the play button will allow players to advance. However, here players will advance to the requirements phase instead of proceeding straight to the main game.

3.2 Requirements Phase

The requirements phase allows players to pick their individual goals for the game. These requirements are sets of individual objectives that award points and other bonuses to the player when completed. They represent the requirements that would be given to a developer or team by a customer for a new software project. Each requirement has three objectives to complete during the game. Each objective is for one of the sprints. Completing an objective by the end of the game will award the player bonus points that will be added to their instruction score and used to determine the winner. If a player completes an objective before the end of the sprint it is associated with they will recieve an additional bonus. These bonuses will not be points, but instead give the player a useful card or status effect. Some sprint bonuses may be awarded immediately upon completing the objective and others may be awarded at the end of the sprint, based on how they will help the player.

When players start the requirements phase they will be taken to a new page that has the available requirements represented as cards. These cards will be on the top portion of the screen and can be scrolled through and selected to see what their objectives are. The bottom portion of the screen will give the conditions and bonuses granted for their completion. When a requirements card is selected it's details will appear here. These details will indicate the sprint the objective is related to, the bonus points for completing the objective, and the bonus given for completing the objective before the end of the sprint. The current prototype for this can be seen in figure???. In the future a more diverse set of requirements will be added as well as specific card art for each requirement.

[2] The Object Management Group (OMG) is a computer industry standards consortium.



Information Security

It is important to secure your system from hackers and malware. You will try to ensure that you protect yourself and ensure that your system is clean

Backlog

1. Played 2 safety cards total, Sprint Bonus: Variable6 Card, End Game Bonus: 10pts
2. Played 4 safety cards total, Sprint Bonus: Redraw2 Card, End Game Bonus: 10pts
3. No Malware or Hack effects at end of game, Sprint Bonus: None, End Game Bonus: 30pts

Figure 1: The Requirements Phase Screen.

Once a player is satisfied with the requirements they have selected they can advance to the planning phase where they will customize a deck to help them complete their requirements. However, an option will exist for players trying out *Agile* mode, or a new requirement, for the first time to pick a pre-built deck. If all human players pick pre-built decks the planning phase will be skipped.

3.3 Planning Phase

Since each of the requirements has different objectives it will be necessary for players to make some choices about how they can best complete these objectives. The planning phase represents the part of a software project where developers make decisions about how the software will be built and what tools they will use. In the planning phase the player will get a base set of cards for their deck. These cards are those that are essential for playing “Game Name”, such as `Instruction` and `Repeat` cards. In addition to this they will be given a pool of cards that they can choose a number of additional cards from. The card pool will be unique to each of the requirements so that players will have access to certain cards that are most useful for that requirement. However, it will also have some cards that may not be essential, but a player may still want as part of their strategy. More powerful cards will be limited in number and will not appear in the card pools for all requirements.

When players start the planning phase they will be taken to a new page where they can build their deck. On a portion of left side of the screen there will be a vertical list of card types. Each type will have a pile showing how many cards of that type, and value if applicable, will be included in the deck automatically. This way the player knows what cards are already in their deck and can act accordingly. The rest of the screen will be split into two horizontal lists of cards. On the top will be the list of cards to add to the deck, and on the bottom the pool of cards the player can pick from. Cards can be dragged between these two lists to move them around. The upper list will have an indicator of how many cards have been added and how many can be added total. Once a player has added the maximum number of cards to this list they will be able to advance to the Implementation phase. If there are more than one human players they will each be given a turn to build their deck.

3.4 Implementation Phase

The implementation phase is the name for the actual game in *Agile* mode. For the most part it play will be the same as it is in standard mode. The major change is that the game no longer ends when a player reaches a specific point total. Instead the players each get an equal number of turns through three 10 turn sprints. Once these sprints are over the game will end. The other major change to the game play is that completing your requirements is now a key part of winning the game. In standard mode it is possible to ignore the bonus objectives and win the game. Generally, the player that reaches the point total first will win. In *Agile* mode the players will need to complete at least some of their requirements in order to win the game. The sprint bonuses will also be useful in making it easier to get more points or to complete subsequent objectives before the end of the game. However, there may be a set of requirements that focuses on the core aspects of the game and is less reliant on the bonuses. This will allow beginner players an opportunity to get used to the format without needing to excell at it immediately.

There will be some small adjustments to the game page’s UI for this mode. The first is that the score limit indicator at the top of the page will be replaced with one that shows the current sprint and how many turns are left in it. When a sprint is over an idicator will

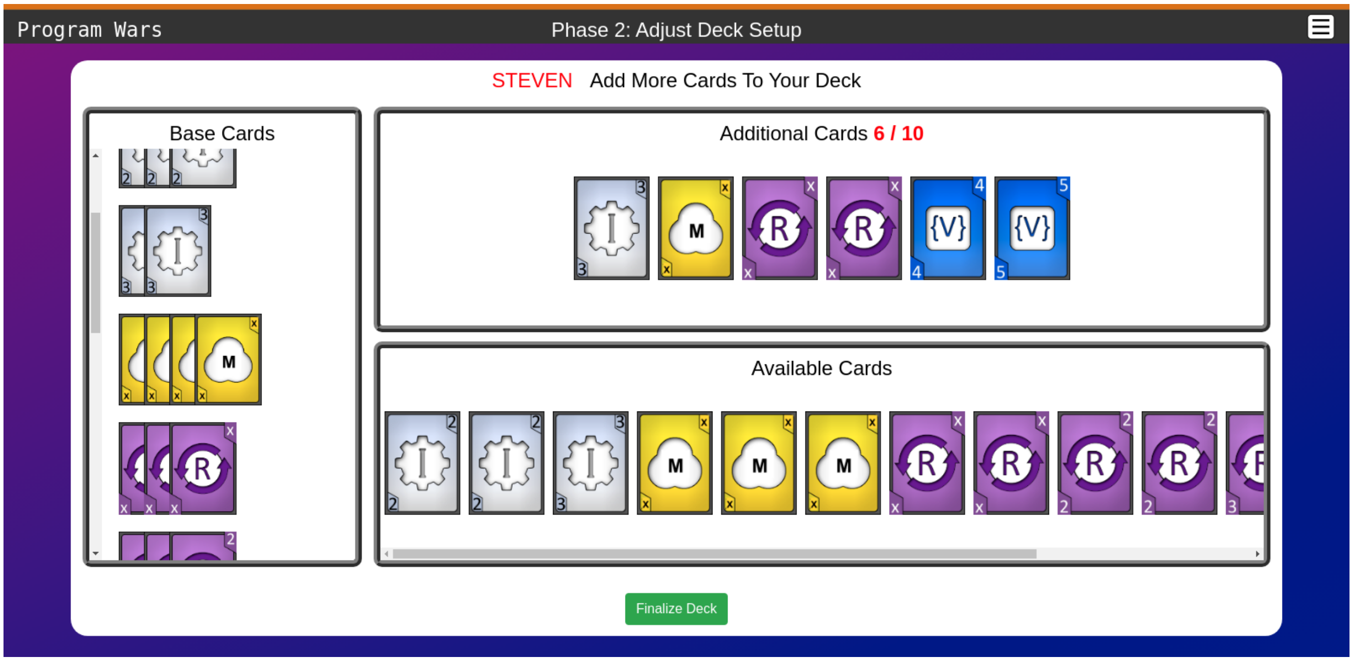


Figure 2: The Planning Phase Page.

be given and all end of sprint bonuses that are due will be given. All other bonuses will be given when they are completed or at the end of the game.

Second the player information panel will no longer need a score meter. Since there is no points total to progress toward the score meter and the current score display will not be useful. They will be replaced with a simple indicator showing the players instructions score. It will also be useful to have some kind of compact indicator for the players progress toward the current sprint's objective. For example, if the player's objective is to build two card stacks that both have nested loops[3] the indicator could say "Nested Loops: 0/2" and update as they built these stacks. This would reduce the context switching the player would need to do during the game to keep track of their requirements.

Lastly, in order to allow the player to see a more detailed view of their set of requirements, a new tab will be added. This replace the standard mode bonus tab. This tab will show a player each of the sprint objectives they need to complete as well as any rewards they will receive for completeing them. Like the current bonus tab awarded bonuses can change color to indicate that they have been given to the player. There can also be clear indicators of progress for each objective.

When all three sprints have ended the game will advance to the testing phase.

3.5 Testing Phase

The testing phase is a replacement for the winner's modal that is displayed at the end of beginner and standard modes. This phase represents a kind of acceptance testing phase for the program the player built. The player had requirements to fulfill for the project so they are judged on their progress. This is where the bonus points for requirements will be added to a player's instruction score. The detailed progress for each player toward their requirements will be shown. Here human players will be able to see how close an AI player came to completing their requirements. The player with the most points will be the winner. Tie breaks will favour the player that completed more requirements in total or by the end of the appropriate sprint. The points given for requirements should be balanced to allow strategies that do not focus as much on instruction score to be competetive if they are completed.

4 GAME CARDS

The player builds their program using the basic building blocks of instructions, methods and repetition. Also, on their turn, a player can launch a cyberattack at an opponent or prepare their defence. This section describes each of the computer programming, cyberattack and cybersecurity cards in the game.

4.1 Computer Programming

In "Game Name" *v*.1.0 the majority of cards focused on computer programming concepts, and most of these cards carry over into "Game Name" *v*.3.0 .

`Instruction`, `Repeat` *and* `Variable`*:* As in actual computer programs, instructions form the backbone of the program. The `Instruction` card represents a fixed number of instructions (1, 2, or 3) and the player uses this type of card as the basis for gaining points in the game. Players can then use the `Repeat` card, which represents the concept of a loop, to further increase the overall number of instructions in their program. There are three sizes of

[3] A card stack with two `Repeat` cards on it.

`Repeat` cards: 2, 3, and 4. By placing a `Repeat` card on another `Repeat` card, the player can form a nested loop.[4] In addition to the fixed-size `Repeat` cards, there is also a variable `Repeat` card (called `Repeat-X`). By itself, this card acts as a `Repeat-1` card. However, the player can place a `Variable` card on a `Repeat-X` to increases its multiplicative power. `Variable` cards have values of 3, 4, 5 and 6.

`Method`*:* In "Game Name" *v*.1.0, the `Group` card represented the concept of a procedure, function or method in a programming language. However, the user study of "Game Name" *v*.1.0 showed this card to be ineffective in conveying this concept. "Game Name" *v*.3.0 replaces the `Group` card with the `Method` card to address this problem.

The `Method` card acts as a proxy for the contents of the *Method Stack* area, with the player's total score being adjusted accordingly. If a new card is added to the *Method Stack* area, the player's score will be adjusted according to the number of `Method` cards in the *Main* area. As with `Instruction` cards, the player can use `Repeat` and `Variable` cards to increase the effect of a `Method` card.

4.2 Malware

"Game Name" *v*.1.0 represented the malware cyberthreat with a single `Malware` card. In "Game Name" *v*.3.0 , the `Malware` card is replaced with cards that more directly represent four of the most common types of malware: `Spyware`, `Ransomware`, `Virus` and `Trojan Horse`.

Spyware is used to gather and send information to another party without the target's consent. The `Spyware` card represents this same situation in the context of the game. Ransomware is used for collecting a specific points from the targeted player. The `Virus` card is used to reduce the effect of a stack of cards in the *Main* area by reducing the points of a card stack and the `Trojan Horse` card is played against an opponent, as a random card in the opponent's hand which replaced with one that mimics it.

4.3 Hacking

"Game Name" *v*.1.0 contained a single card, `Hack`, that represented an intrusion into a computer system. The effect of the `Hack` card was to remove one of the stacks of cards on an opponent's playfield. "Game Name" *v*.3.0 refines this idea by adding specific cards to represent common ways whereby computer systems are intruded or affected by an intrusion. These four cards provide representations of the effects of four types of system attacks: causing a buffer overflow, cross-site scripting, a denial of service attack (DoS), and injection of malicious SQL code.

The `Buffer Overflow` card prevents an opponent from playing any `Instruction`, `Repeat`, `Variable` or `Method` cards for two turns. The `Cross-site Scripting` card stops a player from playing any algorithm or cyberattack cards for two turns. `Denial of Service` card prevents a player from redrawing new cards at the end of their turn and finally the `SQL Injection` card can be used to slow down the progress of an opponent by reducing the total of the *Method Stack* area by two points.

4.4 Cyberdefense:

"Game Name" *v*.1.0 provided three cards for cyberdefense. Two of the cards were *permanent* cards, meaning that they remained on a player's playfield when played, and were referred to as *Safeties*. The first of these cards was the `Antivirus` card which prevented the `Malware` card from being played on a player. The second of these cards was the `Firewall` card which protected against the `Hack` card. The third card was the `Overclock` card, which combated the `Malware` card by increasing the player's total score by 25%. However, it was observed that the `Overclock` effect didn't match well with real-world cybersecurity concepts and was removed in "Game Name" *v*.3.0 .

"Game Name" *v*.3.0 continues the use of the two safety cards, `Antivirus` and `Firewall`, and adds a new one-time-use cyberdefense card called `Computer Scan`.

The `Computer Scan` card represents the action of a user explicitly scanning all of their files to find any infected items using an antivirus tool. The `Antivirus` card reflects this real-world tool by protecting a player from the effect of any of the malware attack cards. `Firewall` card prevent hack cards being played on the player.

4.5 Algorithms/Library Functions:

The use of algorithms, often from libraries, is an essential part of computer programming. Two key categories of algorithms are searching and sorting, and both of these are introduced in "Game Name" *v*.3.0 .

The `Sort` card allows a player to rearrange the top five (5) cards of the deck into whatever order they choose and the `Search` card allows a player to search for a specific card within the top ten (10) cards of the deck.

5 CONCLUSION

Nee to re-write

6 ACKNOWLEDGEMENTS

Need to re-write

REFERENCES

[1] [n. d.]. Battle Bots v2. https://www.curufea.com/doku.php?id=games:board:battlebots. [Online; accessed 03-March-2020].

[2] [n. d.]. Blockly Maze. https://www.brainpop.com/games/blocklymaze. [Online; accessed 03-March-2020].

[3] [n. d.]. CheckIO. https://checkio.org. [Online; accessed 26-Aug-2020].

[4] [n. d.]. Code Master. https://www.thinkfun.com/products/code-master. [Online; accessed 03-March-2020].

[5] [n. d.]. CodeCombat. https://codecombat.com. [Online; accessed 26-Aug-2020].

[6] [n. d.]. CodeWars. https://www.codewars.com/. [Online; accessed 26-Aug-2020].

[7] [n. d.]. CodinGame. https://www.codingame.com/start. [Online; accessed 03-March-2020].

[8] [n. d.]. Kodable. https://www.kodable.com/. [Online; accessed 03-March-2020].

[9] [n. d.]. Potato Pirates. https://potatopirates.game/. [Online; accessed 03-March-2020].

[10] [n. d.]. Program Wars. https://program-wars.firebaseapp.com/. [Online; accessed 03-March-2020].

[11] [n. d.]. Robocode. https://robocode.sourceforge.io/. [Online; accessed 26-Aug-2020].

[12] [n. d.]. Robot Turtles. http://www.robotturtles.com. [Online; accessed 03-March-2020].

[13] Kohl Bromwich, Masood Masoodian, and Bill Rogers. 2012. Crossing the Game Threshold: A System for Teaching Basic Programming Constructs. In *Proceedings of the 13th International Conference of the NZ Chapter of the ACM's Special Interest*

[4] "Game Name" only allows nesting up to two levels to reduce the gameplay complexity and to keep scores from growing too quickly.

Group on Human-Computer Interaction (CHINZ '12). ACM, New York, NY, USA, 56–63. https://doi.org/10.1145/2379256.2379266

[14] Peayton Chen, Rita Kuo, Maiga Chang, and Jia-Sheng Heh. 2009. Designing a Trading Card Game as Educational Reward System to Improve Students' Learning Motivations. *T. Edutainment* 3 (08 2009), 116–128. https://doi.org/10.1007/978-3-642-11245-4_11

[15] Crispin Cowan, Calton Pu, Dave Maier, Heather Hintony, Jonathan Walpole, Peat Bakke, Steve Beattie, Aaron Grier, Perry Wagle, and Qian Zhang. 1998. StackGuard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks. In *Proceedings of the 7th Conference on USENIX Security Symposium - Volume 7 (SSYM'98)*. USENIX Association, USA, 5.

[16] M. R. De Almeida Souza, L. Furtini Veado, R. Teles Moreira, E. Magno Lages Figueiredo, and H. A. X. Costa. 2017. Games for learning: bridging game-related education methods to software engineering knowledge areas. In *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*. 170–179. https://doi.org/10.1109/ICSE-SEET.2017.17

[17] Withheld for blind review. [n. d.]. Withheld for blind review.

[18] Lawson Harold "Bud" Jacobson, Ivar, Pan-Wei Ng, Paul E. McMahon, and Michael Goedicke. 2019. *The Essentials of Modern Software Engineering: Free the Practices from the Method Prisons!* Association for Computing Machinery and Morgan and Claypool.

[19] Yasser Khazaal, Anne Chatton, Roberto Prezzemolo, Fadi Zebouni, Yves Edel, Johan Jacquet, Ornella Ruggeri, Emilie Burnens, Grégoire Monney, Anne-Sylvie Protti, Jean-François Etter, Riaz Khan, Jacques Cornuz, and Daniele Zullino. 2013. Impact of a board-game approach on current smokers: A randomized controlled trial. *Substance abuse treatment, prevention, and policy* 8 (01 2013), 3. https://doi.org/10.1186/1747-597X-8-3

[20] A. Kieyzun, P. J. Guo, K. Jayaraman, and M. D. Ernst. 2009. Automatic creation of SQL Injection and cross-site scripting attacks. In *2009 IEEE 31st International Conference on Software Engineering*. 199–209.

[21] Michael A. Miljanovic and Jeremy S. Bradbury. 2017. RoboBUG: A Serious Game for Learning Debugging Techniques. In *Proceedings of the 2017 ACM Conference on International Computing Education Research (ICER '17)*. Association for Computing Machinery, New York, NY, USA, 93–100. https://doi.org/10.1145/3105726.3106173

[22] Thomas Nooning. 2002. Lock IT Down: Use Libsafe to secure Linux from buffer overflows. https://www.techrepublic.com/article/lock-it-down-use-libsafe-to-secure-linux-from-buffer-overflows/. [Online; accessed 13-August-2020].

[23] Mikki H. Phan, Jo R. Jardina, Sloane Hoyle, and Barbara S. Chaparro. 2012. Examining the Role of Gender in Video Game Usage, Preference, and Behavior. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 56, 1 (2012), 1496–1500. https://doi.org/10.1177/1071181312561297 arXiv:https://doi.org/10.1177/1071181312561297

[24] David Pickton and Sheila Wright. 1998. What's SWOT in strategic analysis? *Strategic Change* 7 (03 1998), 101–109. https://doi.org/10.1002/(SICI)1099-1697(199803/04)7:23.0.CO;2-6

[25] J. Pieper, O. Lueth, M. Goedicke, and P. Forbrig. 2017. A case study of software engineering methods education supported by digital game-based learning: Applying the SEMAT Essence kernel in games and course projects. In *2017 IEEE Global Engineering Education Conference (EDUCON)*. 1689–1699. https://doi.org/10.1109/EDUCON.2017.7943076

[26] Jonathan Bell Swapneel Sheth and Gail Kaiser. 2011. HALO (highly addictive, socially optimized) software engineering. *In Proceedings of the 1st International Workshop on Games and Software Engineering (GAS '11). Association for Computing Machinery, New York, NY, USA* (2011), 29–32. https://doi.org/10.1145/1984674.1984685

[27] Claudia Szabo. 2014. Evaluating GameDevTycoon for Teaching Software Engineering. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE '14)*. Association for Computing Machinery, New York, NY, USA, 403–408. https://doi.org/10.1145/2538862.2538971

[28] N. Tillmann T. Xie and J. de Halleux. 2013. Educational software engineering: Where software engineering, education, and gaming meet. *2013 3rd International Workshop on Games and Software Engineering: Engineering Computer Games to Enable Positive, Progressive Change (GAS), San Francisco, CA* (2013), 36–39.

[29] Ingo J. Timm, Tjorben Bogon, Andreas D. Lattner, and René Schumann. 2008. Teaching Distributed Artificial Intelligence with RoboRally. In *Multiagent System Technologies*, Ralph Bergmann, Gabriela Lindemann, Stefan Kirn, and Michal Pěchouček (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 171–182.

[30] Carnegie Mellon University. 2018. Alice.org. Online. http://www.alice.org

[31] MIT University. 2018. Scratch Homepage. Online. https://scratch.mit.edu

[32] V. Uskov and B. Sekar. 2014. Gamification of software engineering curriculum. *2014 IEEE Frontiers in Education Conference (FIE) Proceedings, Madrid* (2014), 1–8.

[33] Richard Wetzel, Lisa Blum, and Leif Oppermann. 2012. Tidy City: A Location-Based Game Supported by in-Situ and Web-Based Authoring Tools to Enable User-Created Content. In *Proceedings of the International Conference on the Foundations of Digital Games (FDG '12)*. Association for Computing Machinery, New York, NY, USA, 238–241. https://doi.org/10.1145/2282338.2282385