“Game Name” *v.*2.0

A Game-based Approach to Teaching Programming and Cybersecurity Concepts

Anonymized Author(s)

ABSTRACT

“Game Name” is a web-based card game designed to teach the fundamentals of computer programming and cybersecurity. Unlike previous games, “Game Name” provides an interactive program-language independent learning environment where the goal of the game is not to solve a maze. Based on informal comments and the results of a formal user study, a number of aspects of the game were revised. First, a number of cards were either removed or refined. Also, cards were added to introduce the programming concept of algorithms and the user interface was improved to provide more information about the players and the game state. Finally, two modes of play were introduced to provide players a learning path that allows for a more gradual introduction of programming language and cybersecurity concepts.

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. “Game Name” *v.*2.0 : A Game-based Approach to Teaching Programming and Cybersecurity Concepts. In *Proceedings of ACM Conference (Conference’17).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

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

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

The chapter illustrates the related research work as well as the related games that were developed for Game-Based Learning in SE.

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

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.

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. it for company uses. The initial stage is known as 'Quest', which is a preliminary introduction 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. A simple quest can form a chain of quests, like use-cases or bug fixing. 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. In this process, a balanced environment has been created where players progress through multiple difficulties. The game is designed as a simple plugin to integrate with an IDE[2], such as Eclipse or Microsoft Visual Studio. Making the user comfortable with the working environment is one of the significant contributions of HALO. A gamer is more comfortable in the gaming environment using this technique HALO adopted a context switching free environment. HALO is designed for the software industry, where the different projects have different requirements. Along with that is supports various social aspects like teamwork, project management. 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. It also has hints, which are very helpful for learning.

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. As the project grows, team building is an additional responsibility that needs to be handled. When the project is in the saturation stage, new research works should be promoted so that the new technology can be cultivated. Again some small but significant issues regarding team bonding and employee workload also need to be adequately addressed in terms of good team bonding.

[2]Integrated Development Environment (IDE) is known as a software application that provides comprehensive facilities to computer programmers for software development.

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[3] 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. The researchers state that if the study could be conducted in a broader spectrum (larger dataset), the result of the case study might be eligible for a conclusive outcome.

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.

2.2 Games for Learning Software Engineering

Several games have been created for teaching the basics of programming. However, most of them are platform-oriented (i.e. only work on Windows) or target a specific programming language (e.g. C++ or Java). For learning the basics regarding computer programming, it is essential to understand the underlying logic and conceptual ideas. In this section, several types of games, especially board games, card games, desktop-based games and web-based games, are presented.

2.2.1 Board Games. Board games are a part of tabletop games where different components of the game can be moved on an adjacent surface or board according to some predefined rules [19]. The following are a couple of board games, which teach different aspects of computer programming.

Battle Bots v2 [1] was inspired by the Robo Rally [29] game. The required number of players for the game is between two and twenty. In the middle of the board, there is a repair center. Each player will start the game with twenty-two cards. Each move is divided into two groups: the programming round and the action round. In the programming round, players have to play the movement cards. In this stage, no action card can be played. In action round, there are five-phase where different action cards can be played along with movement cards. The player who survives the match wins the game. The main advantage of the game is that the players have to plan about the moves, which is an important programming concept,

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

as in programming, it is essential to set the goal. A player can design his/her game plan depending on the other players' moves or personal requirements. As a result, a fair amount of calculation is needed, which is very important in the logical aspect of computer programming. The game has no actual programming interface, which is one of the drawbacks of the game.

Robot Turtles [12] is a game to teach kids to code by using simple direction cards to move a specific coloured turtle. The main focus of the game is to get the coloured turtle to the same coloured jewel piece on the board. There are three instructions: forwards, rotate 90 degrees left and rotate 90 degrees right. There are also some obstacles where players might have to use different techniques to overcome the situation, and some unique cards are also introduced like fire cards for shooting. The higher the level, the more complex cards need to be used. Regarding programming, the jump card is the most crucial one. The idea of jump cards is that they can be replaced with a set of instruction cards that can be used repeatedly. There is also a bug card that can undo the immediate move the player has made. Robot Turtles is designed for kids' initial steps in programming, like single statement compilation, which is well executed through a single card movement. The introduction of a jump card is an excellent initial concept of function calls. Bug cards introduce the idea of mistakes in programs.

Code Master [4] is a single person puzzle game that teaches logical problem-solving. There are sixty levels in the game with several challenging levels. The game consists of a map that has six different levels ranging from easy (green) to expert (red). There is also a guide scroll that indicates the order in which the program statements should run. Also, the guide has places for conditional tokens. There are three colours: green, blue and red. For each path, there is a specific token related to the colours. Numerical numbers are represented in the map nodes. The goal is to get to a portal and collect crystals using a limited path and moves. The game focuses on the programming construct of conditional statements (i.e. if-else). Some nodes have a self-loop, which illustrates the concept of a repeat statement. Finally, the shortest path selection is also a focus point. The game requires essential thinking before each move, which is vital for developing logical thinking. Unidirectional and bi-directional edges in a graph can be learned by playing the game.

2.2.2 *Card Games.*

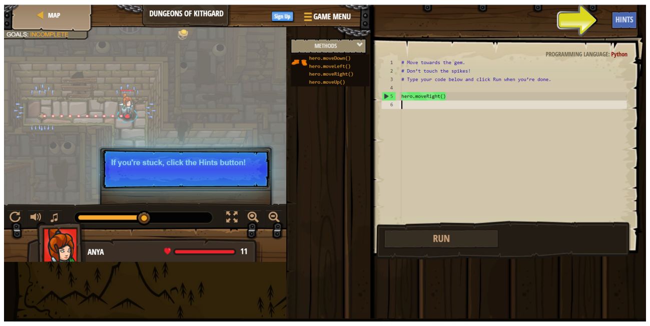
A card game can be defined as a game that involves different playing cards as the main components with which the game can be played [14].

Potato Pirates [9] was the only physical card game found that intends to teach programming concepts. The main objective of the game is to make the user familiar with different programming concepts through social interactions. The final goal of the game is to achieve seven specific cards, named Potato King, through different logical actions performed through different cards. To win the game, a player needs to eliminate all the other ships in possession of the other players or acquire all seven Potato Kings by drawing them from the deck or by eliminating other players' ships and seizing their cards. Players can power up their attacks with programming concepts cards such as loops and conditionals. The game covers the concepts of programmings such as variables, functions, while-loops, if-else conditionals and nested-loops. Some surprise cards serve the purpose of interrupts and control flow. The game illustrates the branching condition very nicely, which is equivalent to the if-else statement. The use of variables is well defined. Different repeat structures are also presented in a reasonable manner such that the game covers the loop structure of programming languages. There is also an option for creating a loop inside a loop, which is known as a nested loop. The game also covers the case structure (switch case or nested if-else).

2.2.3 *Web-based Games.*

Games that can be played on the World Wide Web are known as web-based games. These games use standard web technologies or browser plugins [33].

CodeCombat [5] is a web-based game focusing on learning JavaScript, Python, and other programming languages. The game is built upon concepts of swords-and-sorcery where the player has to role-play as a warrior. The main objective of the game is to learn necessary programming skills by overcoming different challenges like clearing the maze, picking up gems and avoiding any spikes or attacking ogres. The gameplay is split between a code editor on the right and a simulation effect on the left. Figure 1 shows the interface of the game. The avatar is controlled by using a set of commands. In each level, the player has to accomplish a set of tasks. As the game progresses, the player is gradually introduced to new concepts like loops, conditionals, and variables. If a person does not have programming experience, CodeCombat is very suitable. As the game progresses, the tasks involve more complex programming concepts. Most importantly, the levels themselves become more complicated due to possible interactions with the objects in the game world.



```
# Move towards the gem.
# Don't touch the spikes!
# Type your code below and click Run when you're done.

hero.moveRight()
```

Figure 1: CodeCombat gameplay

Blocky maze [2] is a game that introduces the concept of programming loops and conditions in Javascript without writing any Javascript code. The game is a combination of levels that teach programming. It is designed for children who have not had prior experience with computer programming. The game uses a graphical programming language implemented in JavaScript, which can compile to JavaScript, Dart or Python. In the game, programming is done by dragging and dropping code blocks onto a design surface. Figure 2 shows the interface of the game. The main objective of the game is to take the avatar from a starting point to the endpoint. There are a total of ten levels, and the complexity increases as the game progress. Some goals are set to solve the maze in a particular number of steps, which is also challenging. The game interface is similar to a Google Map, which might help the user to adopt the game.

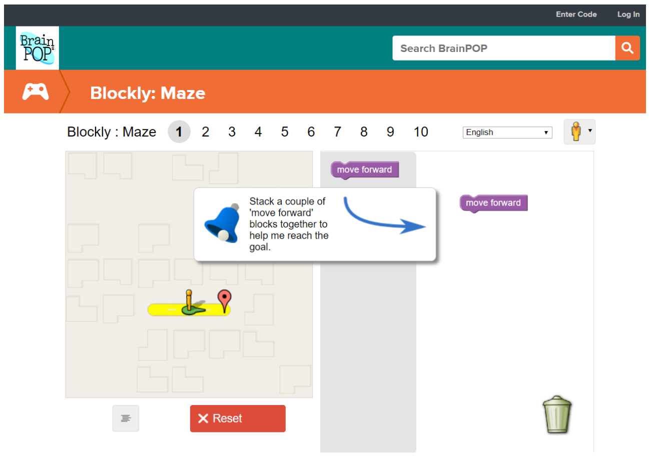
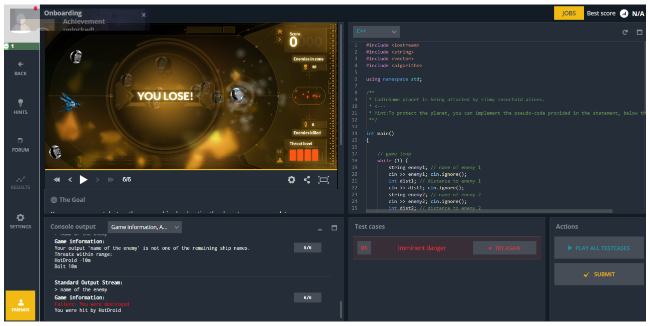


Figure 2: Blockly Maze gameplay

CodinGame [7] supports many programming languages. The main objective of the game is to improve players' coding skills by solving different problems, applying new strategies and getting inspired by other strong opponents. The game can be played in single and multiplayer mode, which gives it the feeling of fun rather than learning. A player can choose any programming language among more than twenty, such as Python, Ruby, Java, and Scala. The targeted group for the game is the people who have basic programming knowledge as well as expert developers. In the game, users create a profile, which is used for challenges and contests. There are opportunities for players to make their profile public so that employers can find them to offer them a job. Also, the game has a forum for members to chat about languages, questions, and share information. The game is not for beginners because it requires some basic knowledge of programming. Figure 3 shows the interface of the game.



```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>

using namespace std;

/**
 * CodinGame planet is being attacked by slimy insectoid aliens.
 * <---
 * Hint:To protect the planet, you can implement the pseudo-code provided in the statement, below th
 **/

int main()
{

    // game loop
    while (1) {
        string enemy1; // name of enemy 1
        cin >> enemy1; cin.ignore();
        int dist1; // distance to enemy 1
        cin >> dist1; cin.ignore();
        string enemy2; // name of enemy 2
        cin >> enemy2; cin.ignore();
        int dist2; // distance to enemy 2
```

```
Game information:
Your output 'name of the enemy' is not one of the remaining ship names.
Threats within range:
HotDroid -10m
Bolt 10m

Standard Output Stream:
> name of the enemy
Game information:
Failure: You were destroyed
You were hit by HotDroid
```

Figure 3: CodinGame gameplay

Kodable [8] is a game developed for kids that focuses on introducing logic and the decisions (i.e. if-then-else) in computer programming. Kodable's programming language introduces players to step-by-step statements with different instructions using in

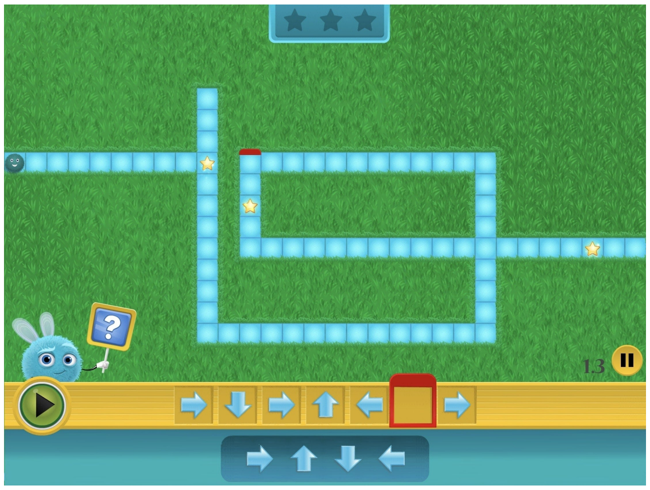


Figure 4: Kodable gameplay

the programming language. The primary programming concepts of conditional statements and loop structure, is well defined in the game. The social aspect of the game is presented into a parents section with written teaching instructions that assist the parent to unlock different levels for kids and extend logic skills into real life. The game is comprised of many features that are appropriate for children. The activities in the game help the player to think like a programmer, solve different problems and eventually to writing real code using the game's custom coding interface. Figure 4 shows the interface of the game. For logic development, the game focus on learning the importance of statement sequence. The game introduces kids to different types of data, such as integers and strings, and data structures, such as arrays. Finally, object-oriented programming concepts are also introduced by the game.

Program Wars [10] is a web-based card game. The game focuses on learning programming and cybersecurity concepts. The player's objective is to achieve a specific number of points through different cards. The game is platform-independent, which means that it is not focused on any single programming language. Instead, the game helps to develop an understanding of the underlying logic of programming. Here the players do not have to be concerned about syntax errors, and programming terms are being described using an elementary vocabulary. In Chapter ??, a more detailed analysis regarding Program Wars is presented.

3 "GAME NAME" *V.*2.0

This section presents a high-level view of the gameplay and a detailed description of the cards that teach programming fundamentals and cybersecurity in "Game Name" *v.*2.0.

3.1 Gameplay Overview

At the start of the game, a player can choose to play either against another human on the same computer (i.e. hot-seat play) or against provided computer opponents. "Game Name" *v.*2.0 expands the original game with two modes of gameplay: *Beginner* and *Standard*. For each gameplay mode, the player can choose to play with one

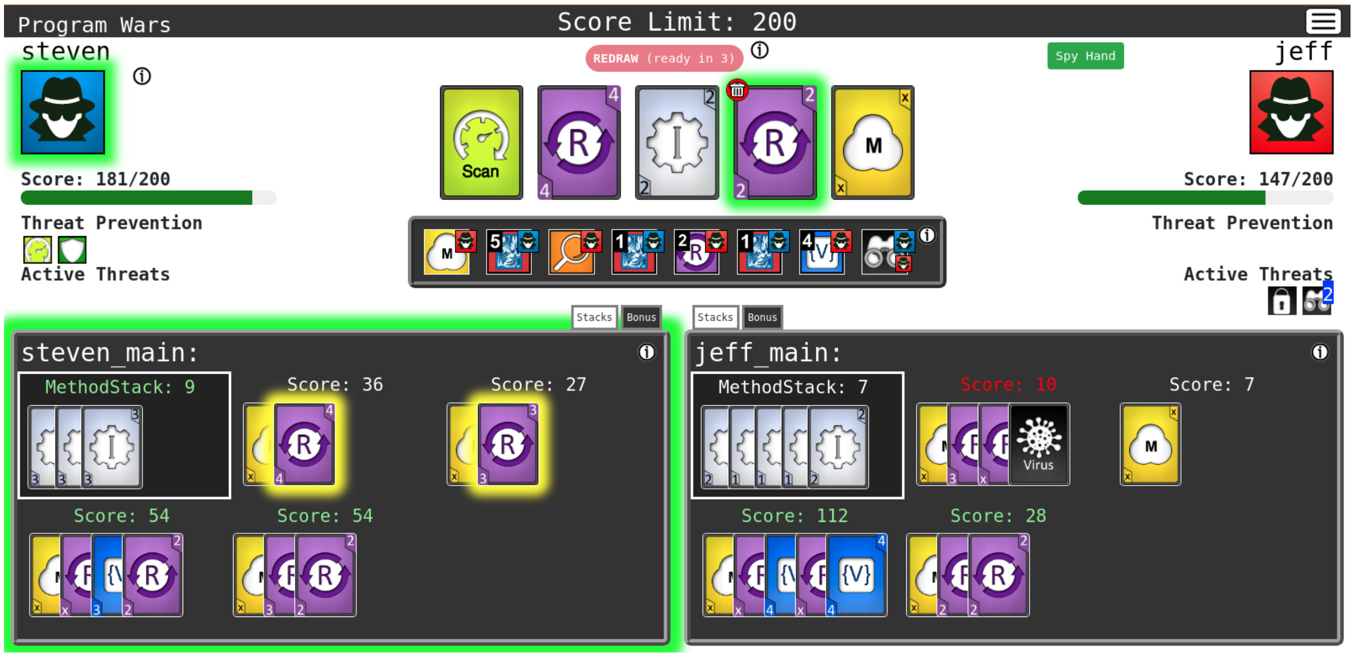


Figure 5: Partway through a game.

of four different sets of cyberattack and cyberdefense cards, adding them to the game's basic deck.[4]

In *Beginner* mode, only one of the two types of cyberattack cards are added to the deck - either two malware cards or two hack cards. In *Standard* mode, the player can choose to play with either all of the malware cards, all of the hack cards, or two different combinations of two malware and two hack cards. Each card set includes the corresponding cybersecurity cards that block the added attacks. These card sets change the gameplay difficulty through the different combinations of cybersecurity and cyberattack cards. The game uses these cards sets to progressively introduce more complicated cards. Table 1 shows the different card sets in each gameplay mode.

At the start of a game, each player is dealt five cards and the player receives another card from the deck at the end of each turn. "Game Name" is played in a series of rounds in which each player takes a turn to either play or discard a card from their hand, or discard and draw a new hand. A player needs to create a program that reaches the goal number of points to win the game. The points represent the total number of instructions that would be executed by the computer-based on the cards in play. Players use a variety of cards to build their program, while also fending off attacks from their opponents.

If either player has reached or exceeded the goal number of points, the game will finish at the end of the current round. This makes it possible for both players to reach the goal number of points in a game. In *Beginner* mode, the player's instruction score is all that is used to determine the winner, and players can tie. In *Standard* mode, bonus points are awarded for reaching specific objectives, such as the use of `Repeat` and `Variable` cards, or ending the game without being under the influence of cyberattacks. If both players have the same total score, these bonus points are used to break ties (i.e. the player who used the most `Variable` cards wins). If this method cannot determine a winner, the players tie.

3.2 Gameplay Areas

Figure 5 shows part of the way through a two-player game. The top right and top left corners of the screen show the status of each player. The player's name is shown along with a unique image that is highlighted with green on their turn. In Figure 5 it is currently Steven's turn. Under each player's image are their score and a progress bar showing their progress towards the goal number of points.[5] Below the player's score is an area to show their current status effects. Status effects are represented by small icons, usually a smaller version of the image from the card that caused the effect. The *Threat Prevention* section shows the current cybersecurity effects active for the player. In Figure 5, Steven has a `Computer Scan` and `Antivirus` effect active. The *Active Threats* section shows any cyberattacks affecting the player. Figure 5 shows that Jeff has been attacked by `Ransomware` and `Spyware`. Effects that last for a specific number of turns are shown with the number of turns remaining over the top right corner. In the example game, the `Spyware`'s effect on Jeff has only two turns until it expires. The status effects are added and removed as cyberattack and cybersecurity cards are played. Further details about cybersecurity and cyberattack cards and their effects are provided in Sections 4.2 and 4.3.

[4]A basic "Game Name" deck is comprised of only `Instruction`, `Method`, `Repeat`, `Variable`, `Search`, `Sort` and `Computer Scan` cards.

[5]The bar is red if the player is below 50% of the goal number of points, yellow if below 75%, and green above 75%.

Table 1: Game modes and Card sets in “Game Name” *v*.2.0.

Type	Card	Beginner				Standard			
		Malware 1	Hack 1	Malware 2	Hack 2	Malware	Hack	Combined 1	Combined 2
Safety	AntiVirus	X		X		X		X	X
	Firewall		X		X		X	X	X
Malware	Spyware	X				X		X	
	Ransomware	X				X			X
	Virus			X		X		X	
	Trojan			X		X			X
Hack	Buffer Overflow		X				X	X	X
	Cross-site Scripting				X		X		
	DoS Attack		X				X		X
	SQL Injection				X		X	X	

Each of the gameplay play areas contains an *Information* icon (an *i* within a circle) that when clicked on provides a short description of that play area.

3.2.1 `Player's Hand`*:* The top of the centre area of Figure 5 shows the current player's hand with the player's currently selected card highlighted with green. When a card is selected, a small trash can icon appears in the upper lefthand corner of the card to allow the player to discard it. Above the cards is a button to allow the player to redraw their hand. If a player redraws their hand, they must wait for three (3) turns to do so again. In Figure 5, the *Redraw* button is inactive and shows that there are three turns until Steven (the current player) can use it again.

Most cards are played by dragging the card from the hand and dropping the card where it is to be added - either the *Main* or *Method Stack* areas of the `Program Editor`. Algorithm, cybersecurity, and cyberattack cards (excluding `Virus`) are played by clicking on them. When these cards are selected, a small overlay appears over the card allowing the player to make a choice. For algorithm cards, the overlay has a single button to activate the card. Safety cards have the same `Activate` button when the card can be activated. However, if the same effect is already applied to the player, then the overlay will indicate that the effect is already active, and the card will not be playable. For cyberattack cards, the overlay will give a set of buttons for valid targets of the attack. The overlay will display “No Targets" if there are no valid targets for the attack. Some cyberattack effects will not allow certain cards to be played. Cards that cannot be played will be highlighted red while in the player's hand and will not be draggable or show an overlay when selected.

3.2.2 `Game History`*:* Below the player's hand is a visual representation of the last eight turns played. Each icon represents a card that was played. In the top right corner of each of these icons is the image of the player that played that particular card. In Figure 5, the leftmost icon shows that Jeff played a `Method` card last turn. If a card had a target player, the target's image is placed on the bottom right corner of the card icon. This can be seen on the rightmost icon in the turn history in Figure 5 where it shows that Steven played a `Spyware` card on Jeff. Some cards may include another card or effect, such as `Computer Scan` when it removes an effect. In this case, a small icon for the card or effect included will be placed on the bottom left corner of the card image (not shown). As the game progresses, the leftmost icon is replaced by the latest card played and all icons are shifted one position to the right.

3.2.3 `Program Editor`*:* The bottom half of the screen represents an editor where the player builds their program. Programs are built by creating stacks of `Instruction`, `Method`, `Repeat` and `Variable` cards. Each stack of cards represents a portion of the player's program. An `Instruction` or `Method` card can be dragged and dropped onto the `Program Editor` to start a new stack. `Repeat` and `Variable` cards are played by dropping them onto an existing stack. A stack with a highlight around the top card indicates that the currently selected card can be played on that stack. If the valid stack is in the current player's `Program Editor`, the highlight will be yellow. For cyberattack cards, such as `Virus`, the valid stack will be in an opponent's `Program Editor`and the highlight will be red. In Figure 5, the currently selected `Repeat` card from Steven's hand can be played on one of the two highlighted stacks.

Each stack shows its score above it telling the player how many points that stack is contributing to the player's score. The *Method Stack* area, which is surrounded by a white border, is a special stack that only accepts instruction cards. The *Method Stack* score is the score used for the `Method` card, is capped at nine (9) points and the stack can hold a maximum of six (6) `Instruction` cards. Placing too many low value `Instruction` cards in a *Method Stack* can leave a player unable to reach the stack's maximum score.

The total score for a player is the sum of all of their stack scores, excluding the *Method Stack*. Stacks with green scores indicate that the stack is complete[6] and no more cards can be added to it. A stack with a red score indicates that the stack is not contributing its full value to the total score. For example, the stack in Jeff's play area with the `Virus` card on top should contribute 21 points (7 for the `Method` card times 3 for the `Repeat-3`). However, the `Virus` card reduces the stack's value by half to 10 (see Section 4.2 for details).

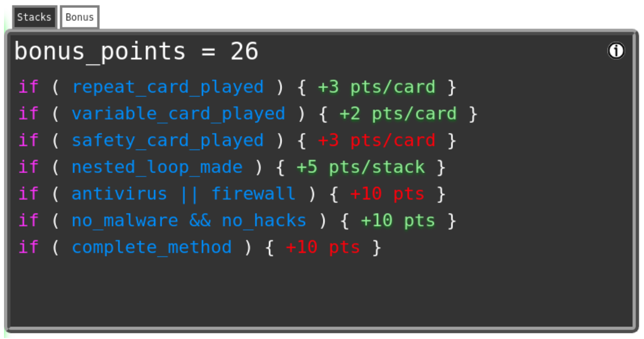
3.2.4 `Gameplay Goals`*:* In *Standard* mode, a player can achieve bonus points by reaching certain goals during a game. A player's

[6]A stack is considered complete when it has two (2) `Repeat` cards played on it. `Repeat-X` cards only count towards completing a stack if they are paired with a `Variable` card.

current bonus points can be seen by switching their play area to the *Bonus* tab. The tabs are in the centre of the screen attached to the `Program Editor` of each player. This tab is not shown for a computer opponent to prevent human players from seeing a computer opponent's bonus progress. The *Bonus* area, shown in Figure 6, has a set of conditional statements written in a C-like pseudocode and replaces the `True` and `False` playfields in "Game Name" *v.*1.0 .

The conditional portion of the statement identifies for what the bonus is awarded. Some bonuses are for playing cards (i.e. `True` when playing a `Repeat` card), and others are for maintaining a certain status (i.e. not being affected by cyberattacks). The body of the condition shows the number of points a player will receive for satisfying the condition. If the text is red, it means the condition is not met, and the text will turn green when the player satisfies the condition. Some conditions, such as the `(no_malware && no_hacks)` condition, may be gained or lost during the game.

At the end of each turn, the player's bonuses are re-calculated and the total bonus score is updated and shown at the top of the *Bonus* tab. Bonus scores are only added to the player's instruction score once the game has finished and do not apply towards reaching the goal score during the game. These bonuses are intended to help reinforce certain concepts and motivate the player towards what can be considered as good programming practices.



```
bonus_points = 26
if ( repeat_card_played ) { +3 pts/card }
if ( variable_card_played ) { +2 pts/card }
if ( safety_card_played ) { +3 pts/card }
if ( nested_loop_made ) { +5 pts/stack }
if ( antivirus || firewall ) { +10 pts }
if ( no_malware && no_hacks ) { +10 pts }
if ( complete_method ) { +10 pts }
```

Figure 6: The *Bonus* tab for a player.

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.*2.0.

`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 `Repeat` cards: 2, 3, and 4. By placing a `Repeat` card on another `Repeat` card, the player can form a nested loop.[7] 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. Figure 5 shows the example where a `Method` card, `Repeat` card and `Variable` card are used.

`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.*2.0 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.*2.0, 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`*:* 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. When a player plays a `Spyware` card against their opponent, a button labelled "Spy Hand" appears beside the opponent's name in the top portion of the screen. Clicking this button allows the attacker to be able to see the affected player's hand. The effect lasts for five turns.

`Ransomware`*:* This card's effect reflects the real-world concept of ransomware where an attacker blocks access to a target's files, such as encrypting them, and threatens to publish or delete them unless a ransom is paid. When a player plays a `Ransomware` card on an opponent, the targeted player loses 10 points from their total score and these points are added to the attacker's score. This can result in an opponent's score becoming negative. Unlike real-world ransomware, recovering from this attack is simple, as an affected player can recover their points by either using a `Computer Scan` or `Antivirus` card.

`Virus`*:* A computer virus is a computer program that replicates itself by modifying other programs. 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. If the stack is built on an `Instruction` card, the stack's score becomes 0. If the stack is built on a `Method` card,

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

the stack's score is reduced by 50%. This adds additional motivation for the player to use the `Method` card. This card is the most similar to the `Malware` card from "Game Name" *v.*1.0, where the `Malware` card reduced a player's total score by 25%.

`Trojan Horse`*:* In the real world, a `Trojan Horse` is a computer program that misleads users as to its real intent. When a `Trojan Horse` card is played against an opponent, a random card in the opponent's hand is replaced with one that mimics it. While players can see when a `Trojan Horse` is played on them they cannot tell which card has been mimicked. The actual effect of the mimic card depends on what card is replaced. `Instruction` and `Method` cards add the `Buffer Overflow` effect to the player instead of creating a new stack in the *Main* area. `Repeat` and `Variable` cards add a `Virus` card to the stack where they were added. Cyberattack cards, such as `Denial of Service` or `Virus`, add a `Cross-site Scripting` to the player instead of adding a cyberattack effect to the target opponent. All cybersecurity and algorithm cards add a `Ransomware` to the player instead of activating the expected card.

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.*2.0 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.

`Buffer Overflow`*:* A common system attack is to send data to a program such that a memory buffer overflows and cause program instructions to be overwritten by malicious code, which is then run. In "Game Name" *v.*2.0, the `Buffer Overflow` card prevents an opponent from playing any `Instruction`, `Repeat`, `Variable` or `Method` cards for two turns. During this effect a *Pass* button is added next to the *Redraw* button. This allows a player to skip their turn if they cannot play any cards and do not want to discard one. The concept behind this card's effect is similar to real-world buffer overflow solutions [15, 22].

`Cross-site Scripting`*:* Cross-site scripting is a code injection attack. The attack happens when the victim visits a web page or web application that administers the harmful code [20]. The visited web page or service acts as a carrier to deliver the malicious code to the affected browser. In "Game Name" *v.*2.0, the `Cross-site Scripting` card stops a player from playing any algorithm or cyberattack cards for two turns. This effect also adds a *Pass* button above the `Player's Hand`. The concept behind this card is to make a player familiar with this type of attack by preventing the advantages given by algorithm and cyberattack cards.

`Denial of Service`*:* A Denial of Service (DoS) attack occurs when a computer system connected to a network is intentionally flooded with requests so that the system can no longer handle legitimate requests. In "Game Name" *v.*2.0, the `Denial of Service` card prevents a player from redrawing new cards at the end of their turn. The effect also adds a *Pass* button above the `Player's Hand` and disables the *Redraw* button. The effect lasts for three turns resulting in the player having fewer cards in their hand to choose from until the effect ends.

`SQL Injection`*:* In an SQL injection attack, malicious SQL code is entered into a data field such that the code is run on a backend database. The result of such an attack is to obtain information that was not intended to be disclosed or delete and/or corrupt the data in the database. In "Game Name" *v.*2.0, 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. The effect lasts until removed by a `Firewall` or `Computer Scan` card. The concept behind this card is that of infiltration of a program's method by malicious code.

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.*2.0.

"Game Name" *v.*2.0 continues the use of the two safety cards, `Antivirus` and `Firewall`, and adds a new one-time-use cyberdefense card called `Computer Scan`.

`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. If the player is under the influence of either a malware or a hack card, then the `Computer Scan` card allows the players to choose a single effect to remove. If the player is not under the influence of a cyberattack card, the effect is saved until the player is attacked, at which time the cyberattack is neutralized and the `Computer Scan` effect is removed.

`Antivirus`*:* An antivirus program is a program or set of programs designed to prevent, search for, detect, and remove malware from a computer system. The `Antivirus` card reflects this real-world tool by protecting a player from the effect of any of the malware attack cards. If the player is already under the effect of one or more malware cards, all of the effects are removed when this card is played. Unlike the `Computer Scan` card, the effect of this card is permanent once it is played, thereby protecting the player from any future malware card attacks.

`Firewall`*:* A firewall is a network security device that controls incoming and outgoing network traffic and grants or prevents data packets based on a set of security rules, thereby protecting a computer system from various intrusion attacks. Like the `Antivirus` card, the `Firewall` card reflects this real-world tool by preventing hack cards from being played on the player. Similar to the `Antivirus` card, if the player is affected by any hack cards, these effects are removed. The effect of this card is also permanent once played.

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*.2.0.

`Sort`*:* Sorting is the arrangement of items into an ordered sequence. In "Game Name" *v*.2.0, the `Sort` card allows a player to rearrange the top five (5) cards of the deck into whatever order they choose. This allows a player to control what cards players will draw for the next five turns, three cards for the player, and two for their opponent. When the card is played, an overlay is opened showing the top five cards and the player can drag and drop cards to reorder them.

`Search`*:* Searching is the process of locating a particular element in a given set of elements. In "Game Name" *v*.2.0, the `Search` card allows a player to search for a specific card within the top ten (10) cards of the deck. Playing the card results in an overlay being opened that shows these cards and the player selects one to immediately put into their hand for their next turn.

5 CONCLUSION

This paper presented an evolution of the game elements and concepts introduced by "Game Name" *v*.1.0 [17]. Regarding programming concepts, the `Group` card was changed to the `Method` card to better represent procedures, functions and methods in programs. Also, the the conditional statement concept was made more explicit with the use of gameplay goals. Similarly, the general cybersecurity concepts in "Game Name" *v*.1.0 were specialized in "Game Name" *v*.2.0. Finally, "Game Name" *v*.2.0 introduced the concepts of algorithms through the addition of the `Search` and `Sort` cards. Regarding gameplay, two modes of play (*Beginner* and *Standard*) were introduced to provide a learning path for players, and the user interface was revised to provide more player and game information.

In the near future, we will be conducting a user study to investigate the effects of "Game Name" *v*.2.0 on learning fundamental concepts of computer programming and cybersecurity. Also, the next version of "Game Name" will introduce a gameplay mode focused on teaching the Software Development Lifecycle (SDLC) and the Agile software development methodology.

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 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