Educational Game Design: An Empirical Study of the Effects of Narrative

Chaima Jemmali
Northeastern University
jemmali.c@husky.neu.edu

Sara Bunian
Northeastern University
banian.s@husky.neu.edu

Andrea Mambretti
Northeastern University
mbr@ccs.neu.edu

Magy Seif El-Nasr
Northeastern University
m.seifel-nasr@
northeastern.edu

ABSTRACT

Integrating narrative elements into a game is a key element in designing an immersive experience. Narrative has been hypothesized to improve engagement, motivation, and learning within educational environments. While empirical results have been produced to show that narrative enhances engagement and motivation, its effects on learning were shown to either be insignificant or negative. We, therefore, aim to address the question of how to integrate narrative in a game to improve learning. We address this through the design of *May's Journey*, an educational game that teaches basic programming concepts where a story is integrated. The game design seamlessly integrates learning goals, core mechanic and narrative elements. In this paper, we discuss the game design as well as a study we conducted to compare two game versions, one with rich narrative and the other with light narrative. Results demonstrate that participants who interacted with the rich narrative version had fewer programming errors and increased engagement within the game. We present our contributions in the form of educational design principles for narrative integration supported by our study and results.

CCS CONCEPTS

• **Human-centered computing** → **Empirical studies in interaction design**;

KEYWORDS

Education, Game Design, Programming, Narrative

ACM Reference Format:
Chaima Jemmali, Sara Bunian, Andrea Mambretti, and Magy Seif El-Nasr. 2018. Educational Game Design: An Empirical Study of the Effects of Narrative. In *Foundations of Digital Games 2018 (FDG18), August 7–10, 2018, Malmö, Sweden.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3235765.3235783

1 INTRODUCTION

Effective games are generally viewed to be intrinsically motivating to players [70], which can entice them to learn and acquire knowledge [3]. Fostering the engaging power of games into educational goals can provide powerful opportunities for deeper learning as their interactive characteristics offer the ability of experiential learning [36], thus improving content relevance and increasing intrinsic motivation [23, 55]. While the relationship between learning and engagement in educational games is under ongoing investigation, most research studies agree that engagement enhances and encourages learning [66, 68].

Integrating narrative elements into a game is a key design element for creating an immersive experience [3, 34]. The work on narrative and games is exhaustive with several conferences associated with the topic. With space limitation, we will only discuss relevant definitions and work. Juul defines game narrative as discourse (the telling of a story) containing the existents and events [31]. Jenkins discusses narrative in terms of plot and story, respectively representing a series of events happening and the viewer's mental construction of it. He also identifies two kinds of narratives in games: one relatively unstructured and controlled by the player and the other pre-structured and embedded within the game [30]. Lindley describes a story in a game as a function of three constructs interacting in the play experience; the pre-designed narrative content, the story potential, and the actual unfolding of the story created by the actions of the player [45]. As identified by Malone's seminal work [49], fantasy in the use of narrative is one of the key elements of games that attracts players through its motivational features such as compelling characters, captivating scenarios, and fantastical settings [50]. Narrative has also been identified to support intrinsic motivation, which has been suggested to generate better learning outcomes [14].

The effect of narrative in educational games has been investigated thoroughly in the literature [16, 18, 44, 50]. However, the success of contextualizing learning with narrative has not been shown empirically. In fact, the few empirical studies that compare different educational game versions with a varying amount of narrative have found that while narrative tend to improve engagement, it had either no effects or negative effects on learning [1, 22, 52, 53, 56, 65].

The question that arises is why a correlation between narrative and learning difficult to achieve? Our hypothesis is that the problem lies in the game design as well as the study setup. In some cases, the narrative elements added to the game are unrelated to the subject taught. In others, the story is not integrated with the gameplay. Finally, adding narrative adds content to the game, which makes it longer to play. That additional time should be taken into account when designing the study comparing between the game versions.

Inspired by game engagement theories, we chose design principles that can create engaging educational narrative games, while also have positive effects on players' performance. We applied these principles to the design of *May's Journey*, a game where players navigate an environmental maze and solve puzzles through programming in an Object Oriented-like custom language. We hypothesize

FDG'18, August 7–10, 2018, Malmö, Sweden

ACM ISBN 978-1-4503-6571-0/18/08…$15.00
https://doi.org/10.1145/3235765.3235783

that the game design and the narrative features would inspire students' interest in learning how to program. Using game-log data and questionnaires, we compare participants' behavior and interest toward programming in two versions of our game; Rich Narrative (RN) vs Light Narrative (LN) that differ in the presence and absence of predefined narrative; the in-game stories or histories revealed via simulated book or dialog text. Our results demonstrate that participants engaged in the RN version had fewer programming errors and increased engagement. Our contribution is two-fold:(1) presenting a set of educational game design principles and their application in a game that teaches programming, and (2) empirical evidence of the positive effects of narrative on student engagement and performance.

2 RELATED WORK

Our research focuses on two major points: methodologies for teaching programming and effects of narrative on educational games.

2.1 Teaching Programming

Teaching programming is difficult and several researchers have tackled this problem from many different perspectives during the past couple of decades [25, 59, 63]. However, there is still no consensus on what is the most effective way to teach programming [25, 69, 74, 75]. One popular method is the constructionist approach, including Scratch [61], Alice [13] or Logo Programming [57], where students are invited to freely explore and construct their programs. However, while increasing interest and engagement, previous empirical studies have shown that students who used these tools had either not discovered important programming concepts, like booleans, variables and control flow, or held misconceptions about their function [24, 39, 51].

Despite the lack of a unified approach to teaching programming, one of the most common viewpoints is that teaching the paradigms of the process is more important than the language choice [10, 46, 47, 80]. To eliminate the barrier of syntax [59], block-based programming, such as Scratch [61], Blockly [21] and recently Reduct [5] has become increasingly popular due to its use of intuitive features, such as drag and drop. However, high school students have also identified drawbacks to block-based programming compared to the conventional text-based approach. Specifically, they perceived it as less authentic and less powerful [76]. On the other hand, existing popular languages such as java or C++ are too verbose and therefore not suitable for educational purposes, especially for beginners [59]. Another approach is to use a custom language, e.g., [20, 43]. The advantages of such an approach are that it reduces the complexity of existing languages while keeping the authenticity of programming. It also can be made easy to use by restricting the scope of functionality and emphasizing only a few concepts at a time [10, 27, 37]. Additionally, debugging can be introduced in a simpler way. This is important because debugging is one of the most essential concepts that has been shown to be difficult for beginners [35, 40, 58]. We, thus, chose to develop a custom programming language for our game.

Programming and puzzles both require inquisitive thinking and efficient problem solving. In fact, computing and informatics community has adopted programming as an application of problem-solving skills [15, 40, 54, 59, 64]. Problem based learning (PBL) is not just about solving the problems but rather about using them to increase knowledge and understanding. This approach is successful only if the problems and scenarios are of high quality [78] which makes designing the problems and, in our case the puzzles, a critical process. Puzzles can also be an efficient and effective method to teach code concepts than tutorial based approaches [27, 43]. Results from [5] demonstrate how a puzzle game can teach programming language semantics in a structured environment. In our work, we discuss a game-based approach due to its ability to provide structure while also offering agency and freedom to play. We are using a structured approach, rather than a constructionist approach, to deliver an environment that teaches programming concepts through a game, where game progression presents the structure for the learner to engage in.

2.2 Effects of narratives

Game-based learning holds a great promise for deep and innovative learning opportunities. Researchers have identified how games can be utilized to effectively enhance learning [2, 23, 49, 62, 72]. Games offer experiential learning opportunities [36] and engage learners through intrinsic motivation [4]. Some educational games have shown improvement in learning outcomes [7, 12, 77] as well as student's engagement in learning [28].

The relationship between learning and engagement in educational games is under ongoing investigation. While cases exist in which a correlation could not be established [26], most research studies seem to agree that engagement enhances and encourages learning. Findings presented in [66] provide evidence of a strong correlation between learning and engagement in a game-based environment that teaches scientific inquiry. Results from [68] provide supporting evidence regarding the positive relationship between learning and engagement in game-based learning.

Previous works suggests that incorporating engaging narrative within a learning environment can aid in supporting engagement and increasing student performance [67]. It is also assumed to support intrinsic motivation [49] which in turn can generate better learning outcomes [43].

It should, therefore, be expected that incorporating narrative elements would improve learning and engagement in an educational context. While there are many theoretical work discussing the effects of narrative in educational games [16, 18, 44, 50], there are few empirical studies investigating the effect of narrative on engagement and learning [11, 16]. These studies report engagement but no learning gains by including the narrative. A study investigating the use of narrative in an educational mystery game for microbiology, compared a rich and a minimal narrative condition of the game to a PowerPoint condition with the same content [56]. Results indicated that the rich narrative condition produced the lowest learning gains in comparison to both the minimal and the PowerPoint condition. The latter produced the highest gains among students. However, the authors noted that the limited amount of time allocated for the experiment may have prevented the participants from fully exploring the game's educational content. Another study compared the effect of art style and narrative complexity of a game designed to help players mitigate three cognitive biases to a professionally designed video for the same purpose [52, 53]. Overall, results indicated that there was

no significant difference between the narrative complexity conditions and the training outcome. These findings could be attributed to the way in which the narrative was integrated; only at the beginning and at the end and not during gameplay. Another study showed no significant differences in learning scores between an educational game with different types of narrative and a game with no narrative [1]. In one case, narrative has been shown to encourage off-task behavior within the game which had negative effects on students' performance [65]. Finally, another recent study looking at the effects of narrative in a math game, also found that the story version did not improve students' performance which was in agreement with participants' perception of the influence of the narrative on their math performance [22]. In this case, the story was also added in the beginning and was not related to the game mechanics or educational goal. Further, in the domain of programming education, one study compared student's experience in learning to program using two versions of a programming environment that differed in their storytelling support [33]. Results indicated that while the storytelling version was more motivating to students, both versions had equal effect on the learning outcome.

In this paper, we aim to further investigate this question adopting a structured approach, where we seamlessly integrate story, game mechanics and learning goals. We also address the issue of timing in our study by not fixing an upper time limit for our participants to play the game.

3 DESIGN

May's Journey is a 3D puzzle game in which players solve an environmental maze by using the game's pseudo code to manipulate the game objects. The game is designed to teach the basics of programming, focusing on logic and object oriented abstractions, while asking learners to type simple instructions in the game's programming language. In a fictional programmed game world that lost its balance, the player is asked to help, May, the protagonist who is attempting to solve the mystery behind the broken world in order to fix it.

3.1 Design Principles

The design of the game is inspired by theories related to what makes games fun as well as what makes educational games effective [38, 48]. The design principles below are what we used in an attempt to achieve a cohesive game where narrative, learning and game mechanics are blended together.

- **P1:** The core mechanic of the game conveys the main learning goals. There can be other game mechanics and other secondary goals but the focus is on having the gameplay serve the knowledge to be acquired.
- **P2:** The game is modular; if one level is broken, too complex, etc., changing that level, removing it or breaking it into smaller levels can be done without disrupting the flow of the game or changing any other level.
- **P3:** The game is structured to encourage "controlled exploration". Levels are grouped into areas in which players can explore in any order, giving them freedom of exploration and a sense of agency. However, to unlock a new area, they need to reach a blocking point that is overcome by solving the previous area's puzzles.
- **P4:** The story is related to the learning matter but does not necessarily communicate its learning objectives. In fact, the story conveys a reason to learn and presents a rational explanation for gameplay and its difference from other games.
- **P5:** Every element in the environment supports the narrative. Props, characters, lights etc, are carefully placed to augment and enrich the narrative.
- **P6:** Hints and Help assist the player's reasoning about the problem but do not give away solutions.

According to Koster [38], fun arises from learning and mastering. In his theory, good games consist of preparation, a core mechanic, a range of challenges requiring a range of abilities and skill, which inspired **P1**. In addition, he thinks that the core of most successful games is made of building blocks, deriving **P2** where we recommend to build the game in a modular way or in "blocks" which also makes difficulty adjustment easier. In fact, to increase fun, games have to push the boundary of skill because if the game is too trivial or too difficult it becomes boring [38]. **P3** was inspired from the updated quest design of *CodeSpells* [19], where levels were broken up into areas teaching specific goals, because using an unstructured approach led students to miss important learning points. According to Malone [48], the essential characteristics of good computer games are challenge, fantasy, and curiosity. Every game should have a goal whose attainment should seem uncertain to the players. Again, he thinks that if a player is certain to win or to lose, they will become bored. He believes that the skill being taught should be used as a mean to achieving the goal but not the goal itself and highlights the importance of intrinsic fantasy which means that the narrative is related to the learning objective. Both these points led to the formulation of **P4**. On the importance of curiosity, he defines cognitive curiosity as the motivation to learn independent from the goal created by the game or the fantasy around it which also supports **P1**. As for environmental elements, they stimulate sensory curiosity **P5** [48] and amplify fun [38]. Finally, Malone mentions the importance of self-esteem and how performance feedback should be presented in a way that minimizes self-esteem damage leading to **P6**. The remainder of this section explains how these principles were applied to our design.

3.2 Narrative

The narrative of the game is meant to not only support engagement, but also provide a logical reasoning behind the fact that players need to program to solve their puzzles **P4**. In fact, students lose interest in programming not only because of difficulty but also because of the lack of compelling contexts in which to learn programming [32]. Curiosity in game design has been identified as an element that promotes engagement [17, 49, 60]. To achieve that, we used foreshadowing techniques, [79] revealing parts of the story built on the Hero's Journey trope. The "hero" May, asks the player for help which represents our "call for adventure". Strange events are happening in her world and she got separated from her friend Juno. Her world is made of code, and now, that code is failing, breaking with it paths that the player will need to program and fix. From the beginning, the narrative is situating players' goals and motivations. The "mentor"

is an Oracle named Myr, she is the one that helped May contact the player. She provides help and explanations throughout the game. The appearing "antagonist" are the Micros who are strong entities also capable of coding. The player will learn only parts of the story at a time, as it unfolds through gameplay and exploration. For instance, In a secret room they find a statue of a Micro revealing a worshiped hero amongst the "enemy". In another level, they learn about the existence of a "master code" capable of saving or destroying the world. They also meet a cat, another game character that will become May's "companion" and help her through her journey. Building on a familiar story template in a learning environment may bring an interesting context in which learners can solve their problems [9, 16].

Figure 1: Exploration phase: The main Character interacting with the Oracle

3.3 Structure

The game is structured in two phases: the exploration phase: The player is free to walk around the world, interact with objects, talk to NPCs and collect items (see Figure 1), and the coding phase: Some parts of the levels are unreachable due to several factors, such as objects blocking the path, a broken path, a locked door, etc. At certain points in the level, the player can use their coding device, depicted as red blocks shown in Figure 1 and 2, to pull up a programming interface. Using the interface, they can type in their code to manipulate the objects and continue crossing the level. This structure has been chosen to embed programming within gameplay, rather than have it as a series of tasks to complete one after another. As emphasized in **P1**, programming is the way that players advance, discover new places and fix broken parts of the game world. The game is divided in areas that teach specific goals as explained in **P3**.

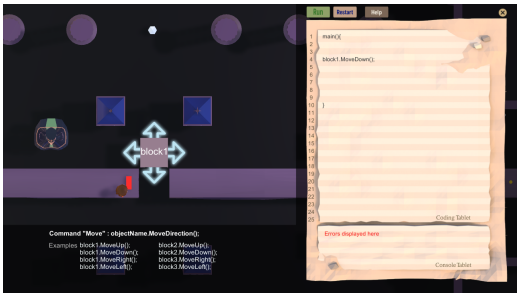

Figure 2: Coding phase: Moving one block in the first level

3.4 Learning Goals

To create easy and relatable analogies, we are choosing to introduce Object Oriented Programming (OOP) following Zhu and Zhou's methodology for teaching the paradigm starting from real world observations [80]. Everything in the game world is presented as an object and every object can be composed of smaller objects. These objects will be introduced through observation and exploration. For example, the Object from class "Block" is any block that a player can interact within the game while Class "windBlock" is a child class that adds certain specificities. Figure 3 shows some of the movable objects in the game. Seeing the pieces of the world as objects and seeing the whole game world as a multitude of programmable objects interacting with each other helps presenting the programming problems as "real life" situations which makes problem solving more efficient [8]. Finally, this design follows Stein's argument for replacing "computation as calculation" with "computation as interaction" [59, 73].

Figure 3: Different types of movable objects in game (from left to right: block, rotatable block, cat, windblock)

3.5 Progression

Currently, the game has 11 levels with different puzzles of varying complexity. In the first stages of the game, we teach basic instructions, sequence logic and loops using methods that are defined within the context. Basic instructions are introduced by two methods: `object.MoveDirection();` where Direction can be Left, Right Up and Down, and `object.Rotate("direction");` where in this case the direction is used as a String argument introducing functions with arguments as well as a new type. Sequence logic is introduced through puzzles that require the player to input commands in a certain order to win the level. Finally, we introduce for loops that take two Integers as begin and end and repeat the commands inside it multiple times. An example code in our programming language would look like the following:

```
main() {
    for (0 to 5) {
        block1.MoveLeft();
        block2.MoveDown();
    }
}
```

For a successful execution, all code must be contained within the main function, which is provided at the beginning of each level or when players press "restart". Our language uses OOP conventions while reducing the verbosity to lower the barrier of syntax.

In each level, we introduce either a new programming construct or a new way to apply a known construct. We are inspired by the Evidence Centered Design presented in [71]. To enable the stealth

assessment of our learning goals, we similarly break our levels into "rule acquisition" and "rule application". Table 1 depicts the breakdown of our levels which, in turn, portrays the application of **P2**.

level	learning goal	rule
level_0	Basic instruction (Move)	acquisition
level_1_1	Basic instruction (Move)	application
	Sequencing	acquisition
level_1_2	Basic instruction (Move)	application
	Sequencing	application
	Pressure plate and Key	acquisition
level_1_3	Basic instruction (Move)	application
	Sequencing	application
	Stair formation and Key	acquisition
level_1_4	Basic instruction (Move)	application
	Sequencing	application
	Block as a moving platform	acquisition
level_2_1	Basic instruction (Move)	application
	WindBlock	acquisition
level_2_2	Basic instruction (Move)	application
	Sequencing	application
	WindBlock	application
level_2_3	Basic instruction (Rotate)	acquisition
	WindBlock	application
level_3_1	Basic instruction (Rotate)	application
	Sequencing	application
level_3_2	For loop	acquisition
	basic instruction (Move)	application
level_3_3	For loop	application
	Basic instruction (Move)	application
	Pressure plate and Key	application
	Cat as programmable object	acquisition

Table 1: List of levels and their corresponding learning objective

P6 of our design principles is implemented through debugging which is introduced by an error handling mechanism that provides a comprehensible feedback. The error messages are designed to be clear, concise and helpful without giving away solutions. A player can make two types of errors: syntax and logical. For syntax errors, the message is displayed in the console. An example can be "Missing opening curly bracket { the for loop in line 5" or "the name ObjectName and/or the command CommandName don't match the commands and objects available". logical errors are presented in a form of a sound cue and a reset of the puzzle to its initial layout.

4 EXPERIMENTS

4.1 Design

We prepared two versions of *May's Journey* varied through the presence or absence of predefined narrative defined by Lindley as the in-game stories or histories revealed via simulated book or dialog text [45]. The differences in the design of the two game conditions (Rich Narrative RN vs Light Narrative LN) are described below. We made sure that the experience of the game itself is not hampered by the removal of narrative elements in the Light Narrative version. The instructions on how to play the game and feedback about errors remain unchanged in both versions and are communicated in the same way through the programming interface.

Figure 4: The intro level of the game showing the hero May asking the player for help

4.1.1 Intro Level. In the RN version, players start by encountering May in a computer desktop environment and are fully introduced to the game's story through text dialog. They are then asked to join her game world to fix it and find her friend. Figure 4 shows part of the story text elements. This scene is removed from the LN version. Removing the opening scene removes the motivation component for the player character and their role in the narrative. This part is crucial since it situates the player's possible motive.

4.1.2 Level 0. This is the introductory level where the player types their first code (Figure 1). Before getting to the programming part, they meet Myr, the oracle who will tell them they need to use programming to rebuild the paths. She mentions that May's task is difficult, but she will be there to help later in the game. In the LN version, the oracle does not appear.

4.1.3 Level 1_4. In both versions, players will find a pressure plate to open a door leading to a manuscript with a secret code. However, only in the RN version, players will discover an old book revealing the existence of a "master code" capable of destroying or saving the world. This information foreshadows a bigger event and builds curiosity in the players' minds increasing their engagement.

4.1.4 Secret Level. This level is only accessible if the player has completed the following steps: activate the pressure plate in level 1_4, get the manuscript in level 1_1, and type the secret code into the right coding button in level 1_3. In the RN version, they will find a statue of the Micro hero, while in the LN version, they only find gems to collect. Revealing a possible antagonist should enhance players' curiosity and build assumptions about the game world history.

4.1.5 Level 3_1. In this level, the player encounters the cat companion. In the RN version, there is a dialog in which May and

the cat mutually agree to help each other. In the LN version, there is no dialog but only a message saying "cat has joined your party". Since the cat can be used as a programmable object, we preferred to keep it to prevent inconsistencies in the gameplay between the two versions. Minimizing the dialog between the two characters reduces the importance of their relationship and future friendship. In the LN, the cat becomes simply a "tool" rather than a companion.

4.1.6 Level 3_2. In the RN version, the players can see an active Micro appearing and disappearing in the distance. The Micro does not show up in the LN. This scene doesn't confront the player with the antagonist but builds their expectation.

4.2 Procedure

For this study, we collected data from 87 participants recruited through Amazon Mechanical turk (MTurk) and limited it to turkers who are located in the United States to limit language issues. MTurk was chosen for recruitment because of its growing popularity amongst researchers looking for a wide range of individuals to generalize their findings [6, 41, 42]. Moreover, we noticed that turkers constitute a good base of users who may not have programming experience.

Participants were asked to complete two questionnaires that included both quantitative and qualitative data. The first one is the pre-game survey, which consisted of fifteen questions to identify user's demographics, game preferences, and prior experience and knowledge of programming. After taking the first questionnaire, participants were asked to play the game for at least one hour or until completion. The game is estimated to take around 45 minutes to finish for a non-programmer. For each participant, we collect game logs of the player actions performed during the gameplay, including interactions with NPCs, objects collected, codes written, and error details. Immediately after completing the game, participants were asked to answer a post-game questionnaire that consisted of twelve questions to determine user engagement and the game's impact on users' programming experience. To measure immersion, flow, competence, positive and negative affect of players, we used the validated Game Experience Questionnaire (GEQ) [29]. To measure participants' opinion about programming, we added programming perception questions to both the pre- and post-game questionnaires that are based on a 5-point Likert scale (1: Definitely Not, 5: Definitely Yes). Participants are asked if they think programming is hard, useful or boring. Participants who completed the post-game survey (they either finished the game or spent at least an hour playing) were compensated with $10.

We first preprocessed and cleaned the data to remove the ones with incomplete or incoherent records. As a result, 14 out of the 87 participants were dismissed. Thus, the experiment was conducted on 73 participants who were randomly assigned to two groups: RN and LN. Table 2 shows the participants' breakdown.

Total (73)			
Light Narrative (34)		Rich Narrative (39)	
Female (12)	Male (22)	Female (18)	Male (21)

Table 2: Participants distribution

5 RESULTS

To understand participants' perception about the game and identify potential differences between them, we examined four primary aspects using our questionnaire data: engagement experience, game elements, participants' interest in future play, and their interest in programming.

5.1 Engagement Experience

We analyzed participants' responses to the in-game GEQ questions that were answered based on a five-point Likert scale with (4: Extremely, 0: not at all). The questions were used to target the overall experience of the game, rather than the programming experience. A Mann-Whitney U-test showed that there are significant differences between the RN and LN groups for some of these measures: "I felt bored" ($p < 0.001$), and "I was interested in game's story" ($p < 0.001$). In analyzing the boredom aspect, we found that 59% of the participants in the RN version extremely agree that they did not feel bored during the gameplay compared to 47% of participants in the LN version of the game. Additionally, 30.8% of the RN participants were extremely interested in the game's story versus 5.9% of the LN participants. Although weakly significant, the answers "I forgot everything around me" ($p = 0.081$) and "I felt completely absorbed" ($p = 0.11$) correlate with participants' interest in the story as shown in the Figure 6. This indicates that RN may have increased immersion.

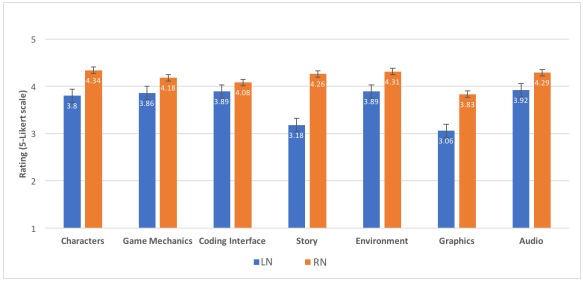

Figure 5: Mean of 5-Likert scale rating on game elements

5.2 Game Elements

To provide insight on how users perceived different elements of the game design, we analyzed their responses to the question "Rate how much you liked these items from 1 to 5 (1 being I did not like it and 5 being I liked it)". These elements include: characters, game mechanics, coding interface, story, environment, graphics and audio. As shown in Figure 5, overall, the RN participants had a higher average rating for the design elements than LN participants. A Mann-Whitney U-test showed that there are significant differences between RN and LN for some of these measures: Story ($p < 0.001$), Environment ($p = 0.047$), Characters ($p = 0.046$), and Graphics ($p < 0.001$). By looking at participant's direct feedback to the question "What did you like most of the game?", one reported "The story and strategy mixed with coding". Another participant said, "It looked like fun-good graphics" while another mentioned "The cute character". This reinforced the findings of participants' positive response to most of the game elements.

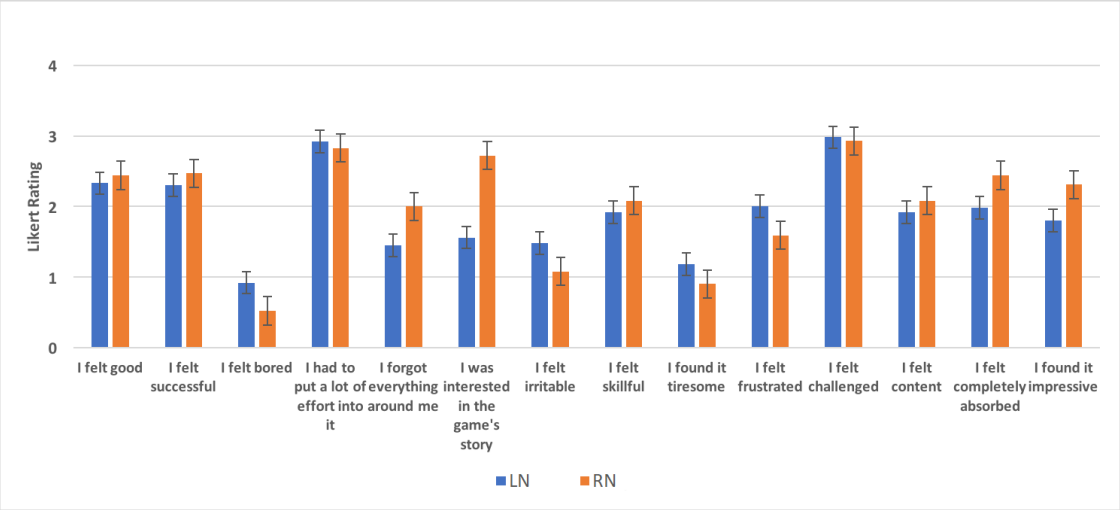

Figure 6: Mean Likert rating of in-game GEQ

5.3 Participants' interest in future play

Based on the analysis of the question "Would you Like to play more of the game?", which is a five-point Likert scale (5: Definitely Yes. 1: Definitely No), there are significant differences between RN and LN participants ($p = 0.037$). We found that 66.7% (Def. Yes: 35.9%, Prob. Yes: 30.8%) of the RN participants had a stronger interest in playing more of the game compared to 47% (Def. Yes: 17.6%, Prob. Yes: 29.4%) of the LN participants. Some participants' responses support these findings. One said "I really wish to keep playing". Another one reported "I wish the game lasted longer". One potential explanation is that a compelling narrative can help retain players and keep them engaged with the learning subject.

5.4 Participants' interest in programming

To evaluate the effectiveness of the game in piquing participants' interests in programming, we examined the perception questions in both the pre- and post-game questionnaires. The questions were based on a five-point Likert scale (5: Definitely Yes. 1: Definitely No). The Mann-Whitney U-test showed that there is significant difference in the question "After playing the game, do you think programming is boring?" with ($p = 0.016$). By analyzing the data, we found that 48.7% of the RN participants reported that they definitely believe programming is not boring after playing the game compared to 29.4% of the LN participants. To support this finding, we further evaluated the difference between participants' responses to the questions "do you think programming is boring?" and "After playing the game, do you think programming is boring?" in the pre-and post-survey respectively. Our goal is to identify if there is an effect of gameplay on changing participants' perception of programming. We found that 48.7% of the RN participants lowered their scoring scale (i.e. they think programming is less boring after playing the game), compared to 35.3% of the LN participants. This indicates that narrative can support student's engagement in the learning task and spur their interest towards programming education. For both of the game versions, we couldn't find any significant differences between the pre- and post- responses of programming being either hard or useful. The majority of participants in both group thought programming became easier after playing the game.

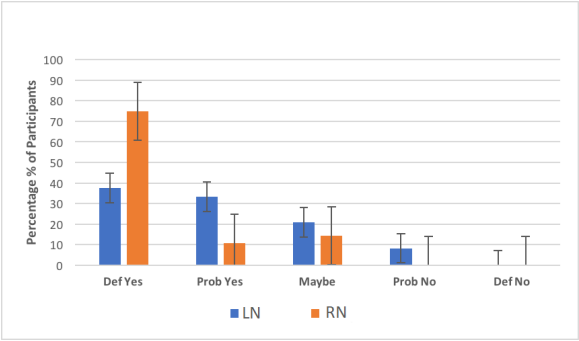

Figure 7: Responses of participants with no/bad programming experience to the question "Do you think the game can be an effective way to learn programming?"

To investigate the effect of the game in easing the learning process for individuals with no prior programming experience, we filtered the data into two groups based on their response to the pre-questionnaire question "How do you estimate your programming experience?". The first group contained 52 participants with no or bad programming experience, in which 24 of them played the LN version whereas 28 played the RN version. The other group was composed of 21 individuals with medium or good programming experience, 10 played the LN version versus 11 for the RN version. Within the same group of participants with no or bad programming experience, we analyzed their responses to the post-game question of "Do you think this game can be an effective way to learn programming?". Results from Figure 7 show that 75% of the participants in

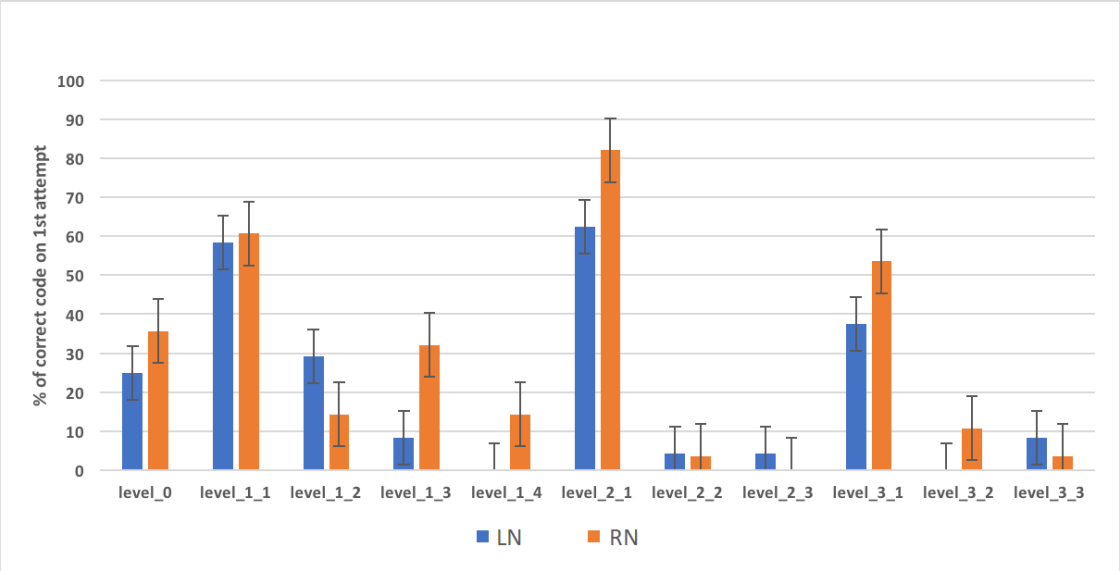

Figure 8: Percentage of individuals with no/bad programming experience that correctly solved the puzzle on their 1st attempt

the RN version think the game is definitely effective to learn programming versus 37.5% for the LN version. This demonstrates that leveraging narrative within a game-based learning environment can be an effective tool to teach programming education to individuals with no prior experience. This is supported by the direct feedback of the participants. One participant reported that "I feel like I'm a better coder than I was an hour ago." Another one stated "the game taught coding in an easy to understand manner."

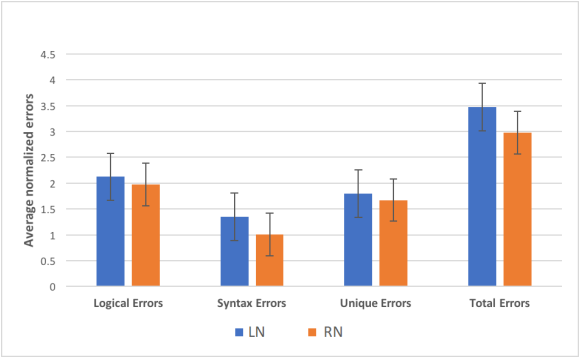

Figure 9: Distribution of the different types of coding errors

5.5 Game Data

To gain better insight into the effect of the game on engagement and programming learning, we analyzed the behavioral data collected from the game logs. Results showed that 69.2% of the RN participants reached the secret level compared to 58.8% of the LN participants. This demonstrates that the rich narrative encouraged participants to engage in extra activities within the game. Also, 56.4% of the RN participants finished the game versus 50% of the LN participants.

As mentioned earlier, the game differentiates between error types and address them in order with a priority for syntax ones. This means that if a code contains both logical and syntax errors, it will be counted as the latter. In addition, each successive set of errors with exactly the same input code are counted as one utterance, and presented as a unique error. To generalize the measurements across participants, we normalized the number of errors over the total coding time spent. As can be seen in Figure 9, across all the error types logged in the game, RN participants had a lower average than LN participants. However, due to a high variance in the number of errors, we could not find significant results across the error types.

To further investigate the effect of the game on programming, we wanted to identify the percentage of participants who correctly solved the coding puzzles on their first attempt. We compared between the two groups of participants having no/bad programming experience and medium/good programming experience. Figure 8 shows the percentage of players with no programming experience who correctly solved the puzzle on their 1st attempt. Major differences can be clearly perceived among the game levels between the players in the two versions of narratives. For instance, in level 2_1, 82.2% of RN participants completed the puzzle correctly compared to 62.2% LN participants. Also, in the RN condition, 14.3% and 10.7% completed levels 1_4 and 3_2 respectively whereas no player has completed these levels correctly on their first attempt in the LN condition. These are typically levels where players need to apply the learned command in a new challenge. However, the Mann-Whitney U-test did not report any significance, except for level 3_1 ($p = 0.026$) which is the level where players meet the cat. This indicates that players in the Rich Narrative version were potentially more focused and more eager to find and save the trapped cat. As for the other group of good programming experience, we could not perceive any differences between the two game conditions.

6 DISCUSSION AND LIMITATIONS

The results of our evaluations suggest that adding rich narrative elements to a game that teaches programming can actually improve engagement and therefore result in increased learning. In a larger context, this demonstrates the benefits of embedding narrative elements, specifically characters, dialog, and foreshadowing, to an educational context. We believe that the design principles we followed can be applied to other educational games. Narrative games offer different types of interactions and it is important to leverage this advantage and fully merge game mechanics and learning content. Specifically, **P1** and **P4** are the major differences between our game design and previous work design which may explain the different performance results.

In fact, when asked about what did they like most about the game, 54% of the participants in both conditions mentioned coding and learning, while 21% indicated the puzzles and the challenge. These results imply that the game was liked because of its educational value. The additional features such as the graphics, audio, environment and narrative elements can be seen as means to support the learning aspect.

One of the limitations is that the game is short, including only a few levels and therefore does not cover many essential programming aspects, such as variables, boolean logic and arrays. A study of a longer game including these concepts along with introduction to abstraction and classes is needed to effectively assess the learning benefits. Another limitation is the relatively small number of participants, which prevents the generalization of some of our findings.

7 CONCLUSION AND FUTURE WORK

We presented *May's Journey*, an educational puzzle game that teaches programming whose design follows principles inspired from theories related to fun and engagement from games research. The main feature is that the core mechanic of the game serves our educational goal. The game's progression is structured to make sure players discover all the materials needed to develop essential skills while allowing exploration and extra levels to push their skills further. The narrative elements, because of their intrinsic nature, succeeded at engaging our participants and improved their performance in the game.

In future work, we plan to expand the content of the game to include other essential programming concepts. Furthermore, we intend to investigate potential adaptive mechanisms to help learners with higher difficulties. Finally, we think that running studies with middle and high school students will be an important step in assessing the educational value of the game. The experiments will involve transference tests to fully assess the learning outcomes.

8 ACKNOWLEDGEMENTS

This research is supported by NSF AISL (Advancing Informal STEM Learning) Award Id : 1810972

REFERENCES

[1] Deanne M Adams, Richard E Mayer, Andrew MacNamara, Alan Koenig, and Richard Wainess. 2012. Narrative games for learning: Testing the discovery and narrative hypotheses. *Journal of educational psychology* 104, 1 (2012), 235.

[2] Clark Aldrich. 2005. *Learning by doing: A comprehensive guide to simulations, computer games, and pedagogy in e-learning and other educational experiences*. John Wiley & Sons.

[3] Alan Amory. 2007. Game object model version II: a theoretical framework for educational game development. *Educational Technology Research and Development* 55, 1 (2007), 51–77.

[4] Alan Amory, Kevin Naicker, Jacky Vincent, and Claudia Adams. 1999. The use of computer games as an educational tool: identification of appropriate game types and game elements. *British Journal of Educational Technology* 30, 4 (1999), 311–321.

[5] Ian Arawjo, Cheng-Yao Wang, Andrew C Myers, Erik Andersen, and François Guimbretière. 2017. Teaching Programming with Gamified Semantics. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 4911–4923.

[6] Michael B Armstrong and Richard N Landers. 2017. An Evaluation of Gamified Training: Using Narrative to Improve Reactions and Learning. *Simulation & Gaming* (2017), 1046878117703749.

[7] Richard Blunt. 2007. Does game-based learning work? Results from three recent studies. In *Proceedings of the Interservice/Industry Training, Simulation, & Education Conference*. 945–955.

[8] David Boud and Grahame Feletti. 1997. *The challenge of problem-based learning*. Psychology Press.

[9] Joseph Campbell. 2008. *The hero with a thousand faces*. Vol. 17. New World Library.

[10] Walter Cazzola and Diego Mathias Olivares. 2016. Gradually learning programming supported by a growable programming language. *IEEE Transactions on Emerging Topics in Computing* 4, 3 (2016), 404–415.

[11] Douglas B Clark, Emily E Tanner-Smith, and Stephen S Killingsworth. 2016. Digital games, design, and learning: A systematic review and meta-analysis. *Review of educational research* 86, 1 (2016), 79–122.

[12] Brianno D Coller and Michael J Scott. 2009. Effectiveness of using a video game to teach a course in mechanical engineering. *Computers & Education* 53, 3 (2009), 900–912.

[13] Stephen Cooper, Wanda Dann, and Randy Pausch. 2000. Alice: a 3-D tool for introductory programming concepts. In *Journal of Computing Sciences in Colleges*, Vol. 15. Consortium for Computing Sciences in Colleges, 107–116.

[14] Edward L Deci and Richard M Ryan. 1985. The general causality orientations scale: Self-determination in personality. *Journal of research in personality* 19, 2 (1985), 109–134.

[15] Giuliana Dettori and Ana Paiva. 2009. Narrative learning in technology-enhanced environments. *Technology-Enhanced Learning* (2009), 55–69.

[16] Michele D Dickey. 2006. Game design narrative for learning: Appropriating adventure game design narrative devices and techniques for the design of interactive learning environments. *Educational Technology Research and Development* 54, 3 (2006), 245–263.

[17] Michele D Dickey. 2011. Murder on Grimm Isle: The impact of game narrative design in an educational game-based learning environment. *British Journal of Educational Technology* 42, 3 (2011), 456–469.

[18] Alejandro Echeverría, Enrique Barrios, Miguel Nussbaum, Matías Améstica, and Sandra Leclerc. 2012. The atomic intrinsic integration approach: A structured methodology for the design of games for the conceptual understanding of physics. *Computers & Education* 59, 2 (2012), 806–816.

[19] Sarah Esper, Stephen R Foster, William G Griswold, Carlos Herrera, and Wyatt Snyder. 2014. CodeSpells: bridging educational language features with industry-standard languages. In *Proceedings of the 14th Koli Calling International Conference on Computing Education Research*. ACM, 05–14.

[20] Matthias Felleisen, Robert Bruce Findler, Matthew Flatt, Shriram Krishnamurthi, Eli Barzilay, Jay McCarthy, and Sam Tobin-Hochstadt. 2015. The racket manifesto. In *LIPIcs-Leibniz International Proceedings in Informatics*, Vol. 32. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[21] N Fraser et al. 2013. Blockly: A visual programming editor. *URL: https://code. google. com/p/blockly* (2013).

[22] Varvara Garneli, Michail Giannakos, and Konstantinos Chorianopoulos. 2017. Serious games as a malleable learning medium: The effects of narrative, gameplay, and making on studentsâĂŹ performance and attitudes. *British Journal of Educational Technology* 48, 3 (2017), 842–859.

[23] James Paul Gee. 2003. What video games have to teach us about learning and literacy. *Computers in Entertainment (CIE)* 1, 1 (2003), 20–20.

[24] Shuchi Grover and Satabdi Basu. 2017. Measuring student learning in introductory block-based programming: Examining misconceptions of loops, variables, and boolean logic. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM, 267–272.

[25] Mark Guzdial. 2004. Programming environments for novices. *Computer science education research* 2004 (2004), 127–154.

[26] Nicole Hallinen, Erin Walker, Ruth Wylie, Amy Ogan, and Christopher Jones. 2009. I was playing when I learned: A narrative game for French aspectual distinctions. In *Proceedings of the Workshop on Intelligent Educational Games at the 14th International Conference on Artificial Intelligence in Education*. 117–120.

[27] Kyle J Harms, Noah Rowlett, and Caitlin Kelleher. 2015. Enabling independent learning of programming concepts through programming completion puzzles. In *Visual Languages and Human-Centric Computing (VL/HCC), 2015 IEEE Symposium on*. IEEE, 271–279.

[28] Leslie J Hinyard and Matthew W Kreuter. 2007. Using narrative communication as a tool for health behavior change: a conceptual, theoretical, and empirical overview. *Health Education & Behavior* 34, 5 (2007), 777–792.

[29] WA IJsselsteijn, YAW De Kort, and Karolien Poels. 2008. The game experience questionnaire. *Manuscript in preparation* (2008).

[30] Henry Jenkins. 2004. Game design as narrative. *Computer* 44 (2004), 53.

[31] Jesper Juul. 2001. Games telling stories? A brief note on games and narratives. *Game studies* 1, 1 (2001), 1–12.

[32] Caitlin Kelleher and Randy Pausch. 2005. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR)* 37, 2 (2005), 83–137.

[33] Caitlin Kelleher, Randy Pausch, and Sara Kiesler. 2007. Storytelling alice motivates middle school girls to learn computer programming. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 1455–1464.

[34] Kristian Kiili. 2005. Digital game-based learning: Towards an experiential gaming model. *The Internet and higher education* 8, 1 (2005), 13–24.

[35] Andrew J Ko, Brad A Myers, and Htet Htet Aung. 2004. Six learning barriers in end-user programming systems. In *Visual Languages and Human Centric Computing, 2004 IEEE Symposium on*. IEEE, 199–206.

[36] David A Kolb. 2014. *Experiential learning: Experience as the source of learning and development*. FT press.

[37] Michael Kölling. 2010. The greenfoot programming environment. *ACM Transactions on Computing Education (TOCE)* 10, 4 (2010), 14.

[38] Raph Koster. 2013. *Theory of fun for game design.* " O'Reilly Media, Inc.".

[39] D Midian Kurland, Catherine A Clement, Ronald Mawby, and Roy D Pea. 1987. Mapping the cognitive demands of learning to program. In *Mirrors of Minds: Patterns of experience in educational computing*. Ablex Publishing Corp., 103–127.

[40] Essi Lahtinen, Kirsti Ala-Mutka, and Hannu-Matti Järvinen. 2005. A study of the difficulties of novice programmers. In *Acm Sigcse Bulletin*, Vol. 37. ACM, 14–18.

[41] Richard N Landers and Tara S Behrend. 2015. An inconvenient truth: Arbitrary distinctions between organizational, Mechanical Turk, and other convenience samples. *Industrial and Organizational Psychology* 8, 2 (2015), 142–164.

[42] Richard N Landers and Rachel C Callan. 2014. Validation of the beneficial and harmful work-related social media behavioral taxonomies: development of the work-related social media questionnaire. *Social Science Computer Review* 32, 5 (2014), 628–646.

[43] Michael J Lee, Faezeh Bahmani, Irwin Kwan, Jilian LaFerte, Polina Charters, Amber Horvath, Fanny Luor, Jill Cao, Catherine Law, Michael Beswetherick, et al. 2014. Principles of a debugging-first puzzle game for computing education. In *Visual Languages and Human-Centric Computing (VL/HCC), 2014 IEEE Symposium on*. IEEE, 57–64.

[44] Cher Ping Lim. 2008. Global citizenship education, school curriculum and games: Learning Mathematics, English and Science as a global citizen. *Computers & Education* 51, 3 (2008), 1073–1093.

[45] Craig A Lindley. 2005. Story and narrative structures in computer games. *Bushoff, Brunhild. ed* (2005).

[46] Paul A Luker. 1989. Never mind the language, what about the paradigm?. In *ACM SIGCSE Bulletin*, Vol. 21. ACM, 252–256.

[47] Paul A Luker. 1994. There's more to OOP than syntax!. In *ACM SIGCSE Bulletin*, Vol. 26. ACM, 56–60.

[48] Thomas W Malone. 1980. What makes things fun to learn? Heuristics for designing instructional computer games. In *Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems*. ACM, 162–169.

[49] Thomas W Malone. 1981. Toward a theory of intrinsically motivating instruction. *Cognitive science* 5, 4 (1981), 333–369.

[50] Thomas W Malone and Mark R Lepper. 1987. Making learning fun: A taxonomy of intrinsic motivations for learning. *Aptitude, learning, and instruction* 3, 1987 (1987), 223–253.

[51] John H Maloney, Kylie Peppler, Yasmin Kafai, Mitchel Resnick, and Natalie Rusk. 2008. *Programming by choice: urban youth learning programming with scratch*. Vol. 40. ACM.

[52] Rosa Mikeal Martey, Adrienne Shaw, Jennifer Stromer-Galley, Kate Kenski, Benjamin Clegg, James Folkestad, Tobi Saulnier, and Tomek Strzalkowski. 2017. Testing the Power of Game Lessons: The Effects of Art Style and Narrative Complexity on Reducing Cognitive Bias. *International Journal of Communication* 11 (2017), 22.

[53] Rosa Mikeal Martey, Adrienne Shaw, Jennifer Stromer-Galley, Kate Kenski, Benjamin A Clegg, James E Folkestad, Emilie T Saulnier, and Tomek Strzalkowski. 2014. Testing the Power of Game Lessons: The Effects of Art and Narrative on Reducing Cognitive Biases.. In *DiGRA*.

[54] Richard E Mayer. 1992. Teaching for transfer of problem-solving skills to computer programming. In *Computer-based learning environments and problem solving*. Springer, 193–206.

[55] Jane McGonigal. 2011. *Reality is broken: Why games make us better and how they can change the world*. Penguin.

[56] Scott McQuiggan, Jonathan Rowe, Sunyoung Lee, and James Lester. 2008. Story-based learning: The impact of narrative on learning experiences and outcomes. In *Intelligent tutoring systems*. Springer, 530–539.

[57] Roy D Pea. 1987. Logo programming and problem solving. (1987).

[58] Roy D Pea and D Midian Kurland. 1984. On the cognitive effects of learning computer programming. *New ideas in psychology* 2, 2 (1984), 137–168.

[59] Arnold Pears, Stephen Seidman, Lauri Malmi, Linda Mannila, Elizabeth Adams, Jens Bennedsen, Marie Devlin, and James Paterson. 2007. A survey of literature on the teaching of introductory programming. *ACM SIGCSE Bulletin* 39, 4 (2007), 204–223.

[60] Eugene F Provenzo Jr. 1991. *Video kids: Making sense of Nintendo.* Harvard University Press.

[61] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, et al. 2009. Scratch: programming for all. *Commun. ACM* 52, 11 (2009), 60–67.

[62] Lloyd P Rieber. 1996. Seriously considering play: Designing interactive learning environments based on the blending of microworlds, simulations, and games. *Educational technology research and development* 44, 2 (1996), 43–58.

[63] Anthony Robins, Janet Rountree, and Nathan Rountree. 2003. Learning and teaching programming: A review and discussion. *Computer science education* 13, 2 (2003), 137–172.

[64] Ralf Romeike. 2008. What's my challenge? The forgotten part of problem solving in computer science education. *Informatics Education-Supporting Computational Thinking* (2008), 122–133.

[65] Jonathan P Rowe, Scott W McQuiggan, Jennifer L Robison, and James C Lester. 2009. Off-Task Behavior in Narrative-Centered Learning Environments.. In *AIED*. 99–106.

[66] Jonathan P Rowe, Lucy R Shores, Bradford W Mott, and James C Lester. 2010. Integrating learning and engagement in narrative-centered learning environments. In *International Conference on Intelligent Tutoring Systems*. Springer, 166–177.

[67] Jonathan P Rowe, Lucy R Shores, Bradford W Mott, and James C Lester. 2011. Integrating learning, problem solving, and engagement in narrative-centered learning environments. *International Journal of Artificial Intelligence in Education* 21, 1-2 (2011), 115–133.

[68] Jennifer L Sabourin and James C Lester. 2014. Affect and engagement in Game-BasedLearning environments. *IEEE Transactions on Affective Computing* 5, 1 (2014), 45–56.

[69] Mara Saeli, Jacob Perrenet, Wim MG Jochems, and Bert Zwaneveld. 2011. Teaching programming in secondary school: a pedagogical content knowledge perspective. *Informatics in Education* 10, 1 (2011).

[70] Katie Salen and Eric Zimmerman. 2004. *Rules of play: Game design fundamentals*. MIT press.

[71] Valerie J Shute, Lubin Wang, Samuel Greiff, Weinan Zhao, and Gregory Moore. 2016. Measuring problem solving skills via stealth assessment in an engaging video game. *Computers in Human Behavior* 63 (2016), 106–117.

[72] Kurt Squire and Sasha Barab. 2004. Replaying history: Engaging urban underserved students in learning world history through computer simulation games. In *Proceedings of the 6th international conference on Learning sciences*. International Society of the Learning Sciences, 505–512.

[73] Lynn Andrea Stein. 1998. What we've swept under the rug: Radically rethinking CS1. *Computer Science Education* 8, 2 (1998), 118–129.

[74] Peter Van Roy, Joe Armstrong, Matthew Flatt, and Boris Magnusson. 2003. The role of language paradigms in teaching programming. In *ACM SIGCSE Bulletin*, Vol. 35. ACM, 269–270.

[75] Arto Vihavainen, Matti Paksula, and Matti Luukkainen. 2011. Extreme apprenticeship method in teaching programming for beginners. In *Proceedings of the 42nd ACM technical symposium on Computer science education*. ACM, 93–98.

[76] David Weintrop and Uri Wilensky. 2015. To block or not to block, that is the question: students' perceptions of blocks-based programming. In *Proceedings of the 14th International Conference on Interaction Design and Children*. ACM, 199–208.

[77] Wee Ling Wong, Cuihua Shen, Luciano Nocera, Eduardo Carriazo, Fei Tang, Shiyamvar Bugga, Harishkumar Narayanan, Hua Wang, and Ute Ritterfeld. 2007. Serious video game effectiveness. In *Proceedings of the international conference on Advances in computer entertainment technology*. ACM, 49–55.

[78] Diana F Wood. 2003. ABC of learning and teaching in medicine: Problem based learning. *BMJ: British Medical Journal* 326, 7384 (2003), 328.

[79] Pieter Wouters, Herre Van Oostendorp, Rudy Boonekamp, and Erik Van der Spek. 2011. The role of Game Discourse Analysis and curiosity in creating engaging and effective serious games by implementing a back story and foreshadowing. *Interacting with Computers* 23, 4 (2011), 329–336.

[80] Haibin Zhu and MengChu Zhou. 2003. Methodology first and language second: A way to teach object-oriented programming. In *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*. ACM, 140–147.