# Reflection in Game-Based Learning: A Survey of Programming Games

JENNIFER VILLAREALE, Drexel University
COLAN F. BIEMER, Northeastern University
MAGY SEIF EL-NASR, Northeastern University
JICHEN ZHU, Drexel University

Reflection is a critical aspect of the learning process. However, educational games tend to focus on supporting learning concepts rather than supporting reflection. While reflection occurs in educational games, the educational game design and research community can benefit from more knowledge of how to facilitate player reflection through game design. In this paper, we examine educational programming games and analyze how reflection is currently supported. We find that current approaches prioritize accuracy over the individual learning process and often only support reflection post-gameplay. Our analysis identifies common reflective features, and we develop a set of open areas for future work. We discuss these promising directions towards engaging the community in developing more mechanics for reflection in educational games.

**ACM Reference Format:**
Jennifer Villareale, Colan F. Biemer, Magy Seif El-Nasr, and Jichen Zhu. 2020. Reflection in Game-Based Learning: A Survey of Programming Games. In *International Conference on the Foundations of Digital Games (FDG '20), September 15–18, 2020, Bugibba, Malta.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3402942.3403011

## 1 INTRODUCTION

Reflection, a cycle of "thinking and doing" [38], is a crucial component of learning [4, 5, 17, 27]. When learners reflect, the otherwise implicit knowledge becomes digested through active interpretation, questioning, and exploration [14, 24]. As games become an accepted media for education and training [17, 19, 34, 35, 42], how well educational games facilitate reflection can have a significant impact on players' learning outcomes.

There is a significant body of work on game-based learning [17, 34, 35], including areas such as educational content generation [20, 29, 43], player engagement and motivation [37, 44], learning outcome [18, 41], and personalized learning [33, 36, 42]. However, only limited work has been dedicated to reflection in educational games [28, 32]. While reflection occurs in educational games, the educational game design and research community can benefit from more principled knowledge of how to facilitate player reflection through game design. For example, a common approach in educational games is to leave guided reflection to post-gameplay assessments or debriefs [16, 45], which usually occur separately from gameplay [45]. However, learning science researchers [14, 24] argue that elaboration and explanation will encourage reflection. Instead of merely letting learners revisit the content, it is more helpful to facilitate their reflection on important issues.

Further, these post-gameplay assessments often provide learners with scores or evaluations to measure the correctness of their gameplay [16, 45]. While this may encourage reflection, the focus remains on accuracy over the learning process.

Reflecting on process is essential to increase learning outcomes and the learner's awareness of their own learning [11, 24, 39]. For example, in computer science education, novice programmers must focus less on the correctness of their solution and more on developing their reflective skills on their process. In this way, they can improve their ability to solve new and unseen problems. Providing players post-gameplay assessments—especially ones that are focused on correctness—may not be enough to motivate reflective behavior at all. Or, if it does, it emphasizes the accuracy of a solution over the learning process [11, 12].

Additionally, due to the diverse work regarding reflection, it is common that researchers and designers work with different definitions that summarize the act of reflection [13]. While these definitions may vary across domains and produce different outcomes, they do not address how reflection is facilitated for a variety of learners or how designers can encourage learners toward this ideal state of thinking. As a result, current design approaches incorporate catch-all features that address reflection as a whole [16, 21, 45]. Often, missing the individual differences that can occur regarding reflection.

More work is needed to understand what reflection means in educational games, how reflection can be facilitated, and what tools currently exist in games that stimulate reflection. Therefore, this paper provides a first step toward identifying common reflective design features used in educational games to develop an understanding of how to better facilitate player reflection through game design.

In this paper, we explore how reflection has currently been supported in educational games designed to teach programming. We use programming games as our domain of analysis because this genre of educational games is especially well-suited to examine reflection due to the demand for conscious decision making and questioning [24], which are crucial aspects of the reflection process [13, 24]. We identified 12 programming games and used them as the basis of our analysis. Our analytical framework contains our prior work on the design space of programming games[47] and two theoretical frameworks on reflection including 1) SchönâĂŹs notions of reflection-in-action and reflection-on-action, to specify when reflection features occur, and 2) Lin et al.'s four reflective design features [24] to specify what reflective elements occur in each game.

This paper presents our analysis of the 12 programming games. Using a close reading approach [3], we identified common reflective design features used in the games. We found that these games provided reflective features; however, their features were often prioritizing the accuracy of a solution after gameplay. We also identify promising directions towards developing a wider range of design features specifically geared towards supporting reflection in learning games.

The rest of the paper is organized as follows. We begin by providing an overview of the literature and existing work on reflection. Then, we follow this by elaborating on our methodology to identify reflective features. We then detail our findings for each feature and conclude the paper with possible directions towards more mechanics for reflection.

## 2 RELATED WORK

In this section, we provide an overview of the literature on reflection and focus on the learning science perspective. Then, we discuss the importance of reflection in Computer Science, specifically, reflective practices when learning how to program due to our focus on programming games. Finally, we pay special attention to existing work on what elements in technology design, including in educational game design, can facilitate and guide reflection.

## 2.1 Reflection and Its Impact on Learning

Reflection has been an active topic in learning science research [4, 5, 17, 27, 38, 39]. Researchers have defined reflection as a process where one can engage in an exploration of experiences that lead to new understandings and appreciations [4]. It has also been described as an essential tool to allow learners to make conscious value choices in their practices [4] and enable them to adapt their thinking to other situations [14, 24]. It is widely accepted that reflection is not only beneficial but also necessary for learning [4, 17, 24, 27, 38, 39]. In reflective practice, a learner becomes aware of otherwise implicit knowledge or behavior [38, 40]. Helping learners to reflect can lead to a variety of benefits, such as self-development, decisions or resolutions of uncertainty; empowerment or emancipation; and other outcomes that are unexpected [27]. Once learners become efficient at reflection, they have more potential to become effective lifelong learners and adapt to new situations [24, 40].

Donald Schön's [38] work in the *Reflective Practitioner* is one of the most widely cited works regarding reflection. He describes two types of reflection, in-action and on-action. Reflection-in-action is the process of revisiting practices during action through "thinking and doing" or testing out a hypothesis in a new environment and using the results to drive the next action. For example, one can be reflective during the action as it is happening, such as playing a song with an instrument and reflecting on how the sound can change. Reflection-on-action is the process of reviewing or analyzing a situation after an event. For example, listening to a recording of the finished song, after the action is done and reviewing it in comparison to other performances. Both of these processes feed into each other, creating a cycle of reflection during and after an event [38]. Reflection-in-action extends thinking into reflection-on-action, and the results motivate the following actions. Therefore, reflection should not be thought of as a separate activity from action, but as an essential part of the overall experience [38, 40]. This framework is particularly useful because it helps identify when reflective elements occur in our game analysis.

## 2.2 Reflection and its Impact on Programming

Reflection is a central skill when learning how to program and has been an active topic in computer science education [11, 12]. Reflective practices are especially important in an environment when there exists more than one approach to solve a problem. For example, CS students must be reflective when finding a viable solution, especially when previous programs developed may not work in a new context. Therefore, analytical skills are needed to better equip CS students on how to go about learning successfully [12].

Edwards [11] explicitly calls for the need to encourage CS students to reflect-in-action when developing code. Novice programmers usually use a "trial and error" strategy when developing programs to see what works and what does not. This behavior is further reinforced by the way current programming assignments in CS education are structured. Assignments tend to emphasize output correctness when students receive feedback only on the end result they produce. Therefore, students equate a program that "produces the right output" with an "effective solution" [11]. Edwards suggests that students must shift to more reflective behaviors where assignments encourage hypothesis-forming and experimental validation to develop students' analytical skills. This issue can also be observed in existing programming games where reflective features, such as post-gameplay assessments, focus on outcomes and the correctness of a solution at the end of each level. To help encourage students to be more reflective in their learning process, Edwards proposes a test-driven development approach that evaluates student performance on how well they have demonstrated the correctness of their program through testing. The goal is to support rapid cycling between writing individual tests and adding small pieces of code, incrementally "thinking and doing." This approach has the potential to increase reflection-in-action in programming games by developing more frequent performance feedback throughout the game experience toward the learning process.

### 2.3 Designing for Player Reflection

Despite its crucial impact on learning, it is generally difficult for learners to be reflective on their own [14]. A common approach in educational games is to display assessments of students learning outcomes. For example, using debriefs to describe important details of the experience, such as a description of events that occurred, a discussion of mistakes, and corrective actions [16, 45]. However, merely revisiting content without elaboration or a reason is not reflection [14]. Some work addresses this issue by asking students to provide explanations for their answers [28, 32]; however, if the answer is incorrect, this may deepen existing misconceptions and may not facilitate reflection toward essential issues at all. It is important for reflectors to be encouraged to think specifically about issues that are considered important [14]. As a result, providing assessments and space for explanation may not be enough to motivate valuable reflective behavior.

To the best of our knowledge, there have been no existing attempts to analyze reflection in educational programming games. Therefore, we widened our scope to look into digital environments that include reflection as part of the design. Previous work addresses reflection in games, such as frameworks that organize reflection into the gameplay cycle as a post-gameplay assessments or debriefs [16, 45], or facilitates reflection through self-explanations, questionnaires [28, 32], and in-game journals [25]. In the context of educational games, several pieces of work discuss how to design for reflection. For instance, Yusoff et al. [45] proposed a conceptual framework for serious games on the basis of nine different principles, with reflection being one [45]. They based this reflection principle on previous work from Garris et al. [16] which discusses using a debrief, a review and analysis of events that occurred in the game, to provide a link between the game cycle and learning outcomes. Yusoff et al. [45] highlighted this work in their framework by creating a separate reflection activity that provides players a description, an explanation of why this activity is chosen, a discussion of the errors made by the learner, and corrective suggestions after gameplay.

Existing work in educational games have focused on using written responses to prompt reflection through explanations. Moreno et al. [28] investigated guidance and reflection in an interactive multimedia game, Design-A-Plant. The goal of their set of studies is to pinpoint the role of guidance and reflection in promoting scientific understanding in agent-based multimedia games for college students. Regarding reflection, in-game agents would prompt an elaborative interrogation, where the agent would ask the student to provide an explanation for their answer during a problem-solving session. Initial results showed that asking students to reflect through explanations on their answers did not affect learning. Moreno et al. [28] suggest that the student's answers could be incorrect and may promote the consolidation of an incorrect mental model by having students verbalize their misconceptions. In a later study, students were asked to reflect on a correct solution rather than their own, which then showed positive results regarding retention and transfer measures.

O'Neil et al. [32] studied the impact of self-explanation prompts in a math game. These prompts are requests for players to type an explanation or to select a reason from a list for each move they made to encourage reflection. This study showed that explanation prompts could help learners make connections between game terminology and mathematics terminology. They found explanation prompts to be an effective instructional method. In some cases, the prompts were less useful when very simple or very abstract questions were asked. Also, the authors do express concern that self-explanation prompts "slow learners down" during the game. This agrees with work from O'Neil et al. which states that prompts were most effective when they were designed to not distract from learning [32].

Additional reflection work can be found in educational tools for teachers. Hughes et al. [21] incorporates a reflective activity called, after-action review (AAR), in a simulated learning environment for teachers. This is a similar approach we see in existing educational games to encourage reflection. AAR provides users time to reflect and guidance with video tagging software and a playback [10, 21]. Additionally, audio and gesture capture accompanies a post-experience assessment of how the user handled different situations during the session [2]. As noted previously, displaying assessments post-experience may not be enough to have learners be reflective on their own, especially toward their learning process. More work is needed on how to encourage and guide learners through the cyclic process of reflection, both in-action and on-action.

Technology design has also explored how we can develop interactive environments to encourage reflective behaviors. The framework we find particularly useful to further the discussion on how to design for reflection in educational games is from Lin et al. [24]. Developed for learning technology in general, this theory discusses how technology can be designed to support reflection, the same objective as this paper, but we narrow our focus to educational games. They describe how reflection can be incorporated in a variety of learning environments and the conditions to facilitate reflection. As a result, this framework is easily transferred to the context of educational games because they provide specific examples that are found in existing technology design. The features include: (1) process displays, (2) process prompts, (3) process models, and (4) social discourse. Each feature will be discussed further below. In addition, Lin et al. [24] state that technology provides a new set of opportunities for reflection, "technology, properly designed and used, enables us to realize reflective learning environments that were not previously possible. (pg.44)" By that logic, games are an applicable domain for new reflective thinking and design.

Another useful framework to further the discussion is Fleck et al.'s [14] Levels of Reflection. These levels incorporate reflection into a behavioral framework by examining purposes of reflection, conditions for reflection, and details of how technologies can support these different levels. The four levels of reflection are R1) Reflective Description, revisiting material; R2) Dialogic Reflection, exploring relationships; R3) Transformative Reflection, revisiting with the intent to do something different; and R4) Critical Reflection, considering aspects beyond the immediate context. We do not use this framework in our analysis because behaviors and activities associated with reflection was the focus. Our work, instead, focuses on design features in programming games that stimulate reflection. However, Fleck et al. make an important argument that we wish to emphasize that we should build technologies to support multiple levels of reflection. To do this, designers should consider various behaviors of their users, which Fleck conveniently organized into levels. Unfortunately, we still do not know how to encourage reflection toward these levels, particularly in games.

## 3 MECHANICS FOR REFLECTION IN GAMES

To advance the knowledge of how to design for reflection, we conducted an exploration of reflection in programming games and what features currently exist that stimulate reflection. We narrow our focus to a sub-genre of this domain, programming games, to start with a smaller sample for our analysis. Additionally, programming games are one of the sub-genres of this domain that are especially well-suited to examine reflection because they tend to embed complex problems that demand conscious decision making and questioning [24]. Following our analysis, we discuss common reflective patterns to investigate open areas for future work in reflection design.

### 3.1 Methodology

In this paper, we utilize Lin et al.'s [24] four Reflective Design Features to identify which reflective elements occur and Donald SchönâĂŹs [38] in-action and on-action framework to specify when these features occur, during gameplay or

| | Blocky:Maze | Cargo-Bot | Code Combat | Code Monkey | Code Spells | CodinGame | Human Resource Machine | Light Bot | Manufactoria | Parallel | RoboZZle | SpaceChem |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Progress Display | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Process Prompts | | | ✓ | ✓ | | ✓ | ✓ | | ✓ | ✓ | | ✓ |
| Process Models | | | | | | ✓ | ✓ | | | | | ✓ |
| Social Discourse | | | ✓ | | ✓ | ✓ | | | | | ✓ | ✓ |

Fig. 1. Reflective Design Patterns in 12 Analyzed Programming Games. A check mark indicates that the reflection feature occurred in the game. Blue: reflection-in-action; Red: reflection-on-action.

post-gameplay. We define each feature and provide examples in section 4. We chose to use these two frameworks due to their focus on design features, which aided in identifying what reflection features occur in technology design and learning environments. Lin et al.'s framework offered specific features found in existing technology that was applicable to transfer to game environments for our analysis.

We select these games based on existing work in educational programming games [47] and on popularity identified with web searches. This led to the selection of the following games: *Blockly: Maze* [26], *Cargo-Bot* [22], *Code Combat* [6], *CodeMonkey* [7], *Code Spells* [30], *CodinGame* [8], *Human Resource Machine* [9], *Light Bot* [23], *Manufactoria* [15], *Parallel* [47], *RoboZZle* [31] and *SpaceChem* [46].

Two researchers developed a shared document that included the feature, its definition, and examples of how this feature was developed in existing technology. All examples and descriptions that were used are listed in the following sections to ensure consistency. Both researchers collectively played each game via a close reading approach [3], where they took notesâĂŤduring and after playâĂŤabout the specifics of what and when reflection features occur. Frequently, comparing their notes and assessments to the feature document. If a feature was unclear, both researchers would interact with the game in question and agree after a discussion. Each game's notes were shared and discussed by both researchers to conclude the results.

## 4 RESULTS

Based on our analysis of 12 programming games, we identify what reflective features are being used and when they occur in each game. Figure 1 shows which games had a particular reflection feature and when they occurred. Next, we define, provide examples, and detail our findings for each feature.

### 4.1 Process Displays

*Process displays* are displays that visualize problem-solving and thinking processes. This feature traces or records students' processes or actions inside the environment. Then, these actions are visualized in the interface that reflects the process back to the students to examine. For example, Geometry Tutor was designed to teach students geometry. The program used process displays to trace, record, and visualize students' actions. The program visually shows the students' action paths in inspectable tree diagrams to allow students to examine their process in more detail [24].

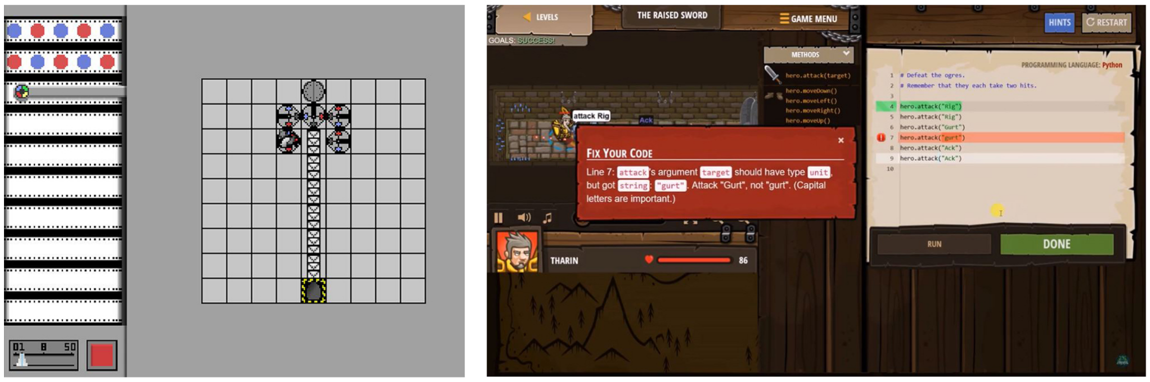
Based on our analysis, 11 out of 12 games provided an in-process playback to visualize the results of their solutions on-action. Often these displays occur on a local, level-by-level basis. All of these games provided a way to visualize the player's program, solution, or creation by playing back their solution in the context of the game's environment and objective. However, these playbacks focuses on output correctness and did not explicitly encourage thinking toward the individual player's learning process. Only one game, *Code Spells* [30], did not use an in-progress visualization or focus on any evaluative outcome; rather, they demonstrated the player's solution in-action in an open world environment. Further, some of these games had a programming space and a game space. The programming space is where the player codes or places components. The game space is where the code can be visualized in the context of the game [47]. We use these spaces to describe the variations of process display.

We observed two ways for process display to occur, either in a split view or in a combined view of the programming and game space. Of the 11 games only 3, *SpaceChem* [46], *Manufactoria* [15], and *Parallel* [47], blended both spaces so the player would program directly onto the game space. Alternatively, the other games all used a split view and had separate spaces, one for programming and another for observing the playback. As a result, they handled the process display feature differently, but still provided a playback for players to reflect-on-action and observe the results of their creations.

For the split view, the player can hit submit to see their solution visualized in the game space. Most of these games use a highlight that steps through the player's code line-by-line. This mechanic is more of a "point" which showcases what code is currently active in the playback. For example, *CodeCombat* [6], is a game about software programming concepts and languages. In this game, players write code to command a hero through a variety of battles. In each level, the players can hit run to play their solution and see their hero enact the written code. As seen in figure 2, the game highlights the code line-by-line to showcase what code is being used by the hero. This visualization encourages reflection-on-action because it steps through the player's process and provides an opportunity to observe the results of each line of code. While this plays, the player can consider their process while observing the results. However, as we will discuss further in the next section, on an error, the playback stops and provides a process prompt to indicate where the player has made an error.

For the combined view, the player could still visualize their solution in the game space; however, there is no highlighting mechanic. After players finish their solution, the game space would show their solution in action. For example, *Manufactoria* [15], as seen in figure 2, the player can drag components onto the track to build directly in the game space. *Manufactoria* [15] provides a play button for players to observe the outcome of their machines in the level's scenario. This encourages reflection-on-action by providing an opportunity to observe their finished machine. Since this view is combined, players can easily cycle between building and observing since they do not have to switch between spaces.

Only one game, *Code Spells* [30], did not use an in-progress visualization. Instead, they demonstrated the player's solution in-action where the player could use their creations in an open world environment. *Code Spells* is an adventure game where players use computer code and programming to act within the game world. Players can create, modify, or edit code in the programming space. They can cast their created spells and navigate the world in the game space, as seen in 3. Both spaces, coding and exploring, potentially encourages more reflection-in-action because the player is active in both areas. This game uniquely incorporates the ability to test out the player's code in the game actively. They can play with and test their spell in a variety of different environments as they explore the game world, thus there are more opportunities to reflect-in-action toward building hypotheses and testing them in both spaces.



FIX YOUR CODE

Line 7: `attack`'s argument `target` should have type `unit`, but got `string`: `"gurt"`. Attack "Gurt", not "gurt". (Capital letters are important.)

```
1  # Defeat the ogres.
2  # Remember that they each take two hits.
3
4  hero.attack("Rig")
5  hero.attack("Rig")
6  hero.attack("Gurt")
7  hero.attack("gurt")
8  hero.attack("Ack")
9  hero.attack("Ack")
10
```

Fig. 2. Manufactoria (Left) uses Process Display by visualizing the player's solution in a playback. CodeCombat (right) uses a Process Display to highlight the player's code, line-by-line and visualize the result of a players code in a playback. This game also uses Process Prompts on failure to encourage reflection on incorrect code.

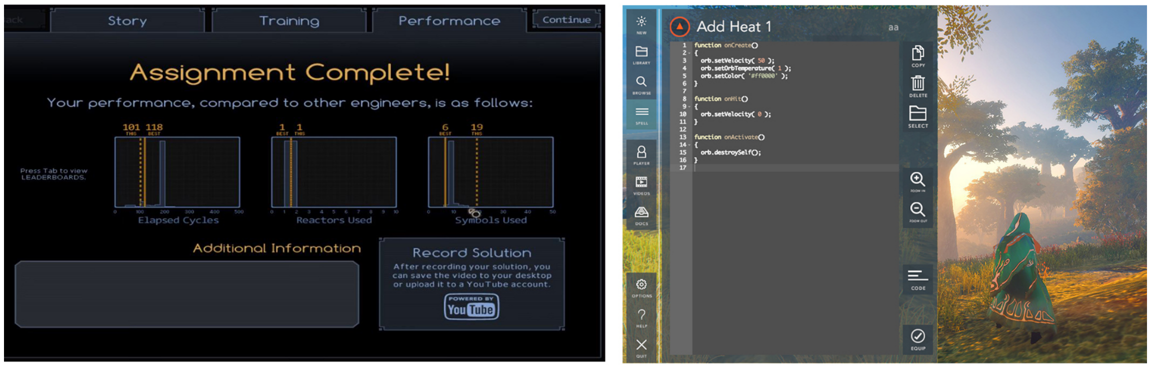
## 4.2 Process Prompts

*Process prompts* are UI elements, such as pop-ups, windows, or pages in a digital interface that poses appropriate questions or content to help guide students in tracking and understanding their own process. Process prompts can be used to encourage reflection by directly asking students questions and requiring them to explain their actions. For a example, Isopod Simulation Program uses process prompts to ask students questions and to encourage students to explain their thought process through a popup window with text. They can also be used to encourage reflection indirectly by helping students generate their own questions through assessments of their work against the given criteria. For an example, CIR-CSIM-Tutor program uses process prompts to aid student reflection on error. This window displays content specific tasks and assesses the student's work against the given criteria [24].

From our analysis, the most common time for a process prompt to occur was on failure. 7 out of 12 games used process prompts to aid student reflection on an error by highlighting specific content in the level, assessing the player against the level's objectives, or displaying helpful tips or corrective content. Since the focus of these prompts were on the correctness of the solution, these prompts drew attention to the level's evaluative outcomes. No prompts were used to guide reflection on the learning process of building a solution. Rather, it highlighted what needed to be fixed before moving to the next level.

We observed two ways that process prompts were used to facilitate reflection-on-action: 1) to a specific area and 2) to the overall goal of the level. *CodeCombat* [6] is a game about software programming concepts and languages. When the player is finished writing their code, the player can hit run to play their solution and see their hero enact the written code line-by-line through a process display feature. However, if the system encounters an error, a popup message appears along side the incorrect code. As seen in figure 2, it uses process prompts to direct player reflection to a specific area that has an error. This popup displays content with corrective details and includes fixes for the player to consider. This prompt encourages reflection-on-action toward a specific area of importance. Once the code is fixed, the player can continue to the next level.

*Parallel* [1, 43] is a game designed to teach parallel programming. In this game, the player needs to coordinate multiple arrows on a track by picking up packages and delivering them to their destination. This game uses process



Assignment Complete!

Your performance, compared to other engineers, is as follows:

After recording your solution, you can save the video to your desktop or upload it to a YouTube account.

```
function onCreate()
{
  orb.setVelocity( 50 );
  orb.setOrbTemperature( 1 );
  orb.setColor( '#ff0000' );
}

function onHit()
{
  orb.setVelocity( 0 );
}

function onActivate()
{
  orb.destroySelf();
}
```

Fig. 3. SpaceChem (Left) uses Process Models by comparing the player's score to other players. Code Spells (Right) uses Social Discourse by incorporating a community library where players can share code.

prompts to encourage player reflection on the level's objectives and goals on a failed solution. When the player submits a solution, the visualization allows the player to observe the incorrect solution play out. If the level goals were not met, the game displays a popup menu with an assessment of their solution against the level's objectives. This encourages reflection-on-action toward the level's objectives.

## 4.3 Process Models

*Process models* are displays that showcase experts' thinking processes so students can compare and contrast with their own process. For example, ASK Jasper, a computer-based learning environment, teaches students basic geometry through designing a swing set. Students can compare and contrast their own designs and reflect on the design process of an expert [24].

Process Models display other expert solutions or actions in a learning environment through pages or other windows. Often these occur after the learner has first completed their solution on their own. We found that 3 out of the 12 games use process models to facilitate reflection on another player's solution through a leader board or comparative stat menus. None of the games incorporated an explicit expert to model a particular process in the game. *CodinGame* [8] used a community voted leader board which allows players to view higher voted solutions. This activity takes place outside the game on a separate web page and does not incorporate this model directly into the game. Even though the game does not display how an expert would solve a level, players can observe other player's code based on what the community voted as the top solution. Since we did not observe any expert models in our analysis, we focus on how players compare and contrast their own process with another to locate where these expert models could potentially go into educational games.

Process models display what others have done at the same level or, more often, include a more distilled version of the solution with a comparative stat menu. For example, *SpaceChem* [46], see figure 3 is a game that teaches concepts related to both chemistry and programming by using a visual programming language. In this game, players build chemical molecules via an assembly line by programming a variety of components. *SpaceChem* [46] uses process models to encourage reflection by using a popup menu with a histogram. It facilitates reflection on the player's stats against the general average. This display allows players to see the average time and number of parts used before they start the level. After a successful solution has been reached, the popup menu appears again to show where the player stands in comparison to others. These do not provide expert answers, but players are provided with a "compare and contrast" reflection opportunity that encourages players to revisit and improve their solution.

### 4.4 Social Discourse

*Social Discourse* is a space in the interface for community-based discussion where students can examine multiple perspectives and receive feedback on their process that can then be used for reflection. This feature refers to the collaborative nature of reflection when a learner seeks input from a group to modify their practice. For example, CSILE is a program for students to collaborate. In this program, students can discuss questions and collaborate to understanding a problem [24].

Social discourse allows learners to participate in discussions and provide feedback on open problems. Additionally, this offers different perspectives on the same problem. We found that 5 out of 12 games use a social discourse feature in a shared content library or in an external website. However, none of the games we analyzed incorporated an area for discussions inside of the game. Only one game, *Code Spells* [30], incorporated a social feature in which players could share code. As seen in figure 3, this game uses a form of social discourse to encourage reflection on other spells through a library player submitted code. It facilitates reflection on different approaches to coding and thinking through a solution. The game's interface provides access to a community library of other published spells. Players can view, edit or reference spells that other players have published. However, there is no direct communication in the game. Reflection occurs on-action by players evaluating other player's code. Even though this game did not include a space for discussion, it displayed an approach to encourage collaboration through participation in a game library.

Another social feature we observed included conversations, such as how others would approach a problem, and questions in which the community can provide answers and feedback to the player. It encourages reflection on multiple perspectives. However, this occurs outside of the game on a website. *CodinGame* [8] uses social discourse to promote reflection on other player's solutions through web pages separate from the puzzle. The website displays voted solutions that form a leader board. They also include a space for conversations under each solution and in a forum in which people can discuss puzzles and provide feedback to other puzzles.

## 5 DISCUSSION

In the previous sections, we present our analysis of 12 programming games and detail what mechanics have been used to stimulate reflection. Based on our observations, all 12 games demonstrated reflective features; however, we observed reoccurring designs that expose a lack of reflection mechanics in educational games.

We observed only one game, *Code Spells* [30], use a reflective feature during gameplay (reflection-in-action), as most of the games provided post-gameplay assessments (reflection-on-action), separate from the gameplay. This disparity is inconsistent with existing learning science theory that argues reflection needs to take place both during and after an action [38]. While considering learning science theories, this can be a useful starting point to encourage new forms of reflection mechanics that have yet to be explored in educational games.

While all 12 games used some form of process display, some games only offered playback of learners' actions. Based on Lin et al.'s work [24], it would be useful for these games to add guided prompts, both in random and fixed presentation schedules, so that learners' attention can be drawn on specific aspects of their learning process. For example, in *CodeCombat* [6], we could add guided prompts to the solution playback as a way to direct attention to specific areas or ask players to elaborate on a solution, as opposed to only prompting the player on error.

Below we describe our observations regarding the common design patterns found from our analysis and suggest open areas for future work. We discuss these promising directions towards engaging the community in developing more mechanics for reflection.

***Supporting reflection-in-action.*** The first pattern we observed from our analysis was that very few games had reflective features during gameplay and often left reflection to post-gameplay assessments or debriefs. As a result, supporting the whole process reflection, especially during gameplay, is a significantly underexplored area in programming games.

This design choice may indicate designers not wanting to overwhelm or interrupt the gameplay experience, as some of the more complex puzzle games already require a significant amount of thinking and doing to complete the level. However, by not considering reflection features during gameplay, players may be more inclined to focus on the accuracy of the solution, or only on getting a correct answer, as opposed to building an effective solution. Reflection support is especially needed in programming games as a solution may work in one setting, but may not always work in a new context. Encouraging more frequent feedback during gameplay may help direct reflection on the learning process.

A possible direction is to consider reflection as its own gameplay loop and how it evolves during and after each level, similar to any other in-game skill. For reflection-in-action, incrementally using process display and process prompts during gameplay may support a consistent reflection cycle and direct thinking toward important areas. We suggest allowing the player to edit their solution while the in-progress visualization is occurring. This may provide the player with a more active role in these playbacks. Process prompts can also be used to support frequent feedback and direction toward what the player may need to reflect on specifically.

For reflection-on-action, a possible direction is to incorporate a social discourse feature to highlight different perspectives on a problem. This feature may emphasize process over outcome, as this can showcase a variety of approaches rather than displaying the average scores as this may further reinforce thinking on accuracy. A social feature can also promote the collaborative nature of reflection when a player examines and shares different approaches in the community.

***Reflection from a global perspective.*** The second pattern we observed from our analysis was that all refection features were confined to local, level-by-level content. None of the features we surveyed provided a higher viewpoint for players to reflect on their actions across multiple levels or problems. This pattern is another underexplored area for reflection work.

By visualizing the player's gameplay over numerous levels, it may provide more opportunities for players to reflect on their behaviors that may otherwise be missed from a local perspective, such as common strategies or approaches. A possible direction is to use a process display feature to showcase gameplay strategies over multiple levels or attempts on a solution. Additionally, providing an opportunity for the player to compare process displays with other players may also emphasize the differences in an approach, further highlighting otherwise unnoticed behaviors.

***Multiple forms of reflection.*** The third pattern we observed was that the reflection features were always consistent for each player. This consistency provides a cohesive player experience but does not address the differences in reflection. For example, a process prompt may encourage reflection in one player, but may not in another. Exploring multiple forms of reflection is another underexplored area in educational games.

Based on work in section 2, reflective behaviors can vary depending on the individual. It is crucial to develop this work further if educational games are to ensure every player gains the benefits of reflection to improve their learning outcome. As a research community, we need to better understand how reflection is facilitated for different people and how games can support these differences.

Existing work by Fleck et al. [13] has already established different levels of reflection. Yet, we still do not know the different ways to facilitate reflection toward these desired levels. A user study is warranted to explore these individual differences and how game design can be used to support reflection further.

## 6 CONCLUSIONS AND FUTURE WORK

In this paper, we explore reflection in 12 programming games and detail what features have been used to stimulate reflection. We identify existing reflection mechanics in programming games and what tools currently exist to facilitate this behavior. As a result of our analysis, we developed a set of common reflective patterns and suggested open areas for future work. We discuss these promising directions towards engaging the community in developing more mechanics for reflection in educational games. A limitation in our work is our small sample of games, and we encourage researchers to extend this work to a larger sample and investigation into other genres of educational games. Finally, we note the subjectivity of our results and discussion, and we support different interpretations of this topic.

## ACKNOWLEDGMENTS

This work is partially supported by the National Science Foundation under Grant Number 1917855. The authors would like to thank all current members of the project.

## REFERENCES

[1] Katelyn Bright Alderfer, Brian K Smith, Santiago Ontañón, Bruce Char, Jessica Nebolsky, Jichen Zhu, Anushay Furqan, Evan Freed, Justin Patterson, and Josep Valls-Vargas. 2018. Lessons Learned From an Interactive Educational Computer Game About Concurrent Programming. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. Proceedings of the 49th ACM Technical Symposium on Computer Science Education, 1077–1077.

[2] Roghayeh Barmaki and Charles E Hughes. 2018. Embodiment analytics of practicing teachers in a virtual immersive environment. *Journal of Computer Assisted Learning* 34, 4 (2018), 387–396.

[3] Jim Bizzocchi and Joshua Tanenbaum. 2011. Well read: Applying close reading techniques to gameplay experiences. *Well Played 3.0: Video Games, Value and Meaning* (2011), 262–290.

[4] David Boud, Rosemary Keogh, and David Walker. 1996. Promoting reflection in learning: A model. *Boundaries of adult learning* 1 (1996), 32–56.

[5] John D Bransford and Barry S Stein. 1993. The IDEAL problem solver. (1993).

[6] Inc CodeCombat. 2014. CodeCombat. https://codecombat.com/

[7] Inc CodeMonkey Studios. 2014. CodeMonkey, programming game. https://www.codemonkey.com/

[8] CodinGame. 2015. CodinGame. https://www.codingame.com/

[9] Tomorrow Corporation. 2015. Human Resource Machine. https://tomorrowcorporation.com/humanresourcemachine

[10] Lisa A Dieker, Charles E Hughes, Michael C Hynes, and Carrie Straub. 2017. Using simulated virtual environments to improve teacher performance. *School University Partnerships (Journal of the National Association for Professional Development Schools): Special Issue: Technology to Enhance PDS* 10, 3 (2017), 62–81.

[11] Stephen H Edwards. 2004. Using software testing to move students from trial-and-error to reflection-in-action. In *Proceedings of the 35th SIGCSE technical symposium on Computer science education*. 26–30.

[12] Alan Fekete, Judy Kay, Jeff Kingston, and Kapila Wimalaratne. 2000. Supporting reflection in introductory computer science. *ACM SIGCSE Bulletin* 32, 1 (2000), 144–148.

[13] Rowanne Fleck and Geraldine Fitzpatrick. 2006. Supporting collaborative reflection with passive image capture. (2006).

[14] Rowanne Fleck and Geraldine Fitzpatrick. 2010. Reflecting on reflection: framing a design landscape. In *Proceedings of the 22nd Conference of the Computer-Human Interaction Special Interest Group of Australia on Computer-Human Interaction*. ACM, 216–223.

[15] PleasingFungus Games. 2010. Manufactoria. http://pleasingfungus.com/Manufactoria/

[16] Rosemary Garris, Robert Ahlers, and James E Driskell. 2002. Games, motivation, and learning: A research and practice model. *Simulation & gaming* 33, 4 (2002), 441–467.

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

[18] Michail N Giannakos. 2013. Enjoy and learn with educational games: Examining factors affecting learning performance. *Computers & Education* 68 (2013), 429–439.

[19] Margaret A Honey and Margaret L Hilton. 2011. Learning science through computer games. *National Academies Press, Washington, DC* (2011).

[20] Danial Hooshyar, Moslem Yousefi, Minhong Wang, and Heuiseok Lim. 2018. A data-driven procedural-content-generation approach for educational games. *Journal of Computer Assisted Learning* 34, 6 (2018), 731–739.

[21] Charles E Hughes, Thomas Hall, Kathleen Ingraham, Jennifer A Epstein, and Darin E Hughes. 2016. Enhancing protective role-playing behaviors through avatar-based scenarios. In *2016 IEEE International Conference on Serious Games and Applications for Health (SeGAH)*. IEEE, 1–6.

[22] Two Lives Left and Rui Viana. 2012. Cargo-Bot. https://twolivesleft.com/CargoBot/

[23] Inc Light Bot. 2008. Light Bot. https://lightbot.com/flash.html

[24] Xiaodong Lin, Cindy Hmelo, Charles K Kinzer, and Teresa J Secules. 1999. Designing technology to support reflection. *Educational Technology Research and Development* 47, 3 (1999), 43–62.

[25] Dennis Maciuszek and Alke Martens. 2011. Computer role-playing games as an educational game genre: Activities and reflection. In *European Conference on Games Based Learning*. Academic Conferences International Limited, 368–500.

[26] Blockly Games Maze. 2018. Blockly: Maze, programming game. https://blockly.games/maze

[27] Jennifer A Moon. 2013. *Reflection in learning and professional development: Theory and practice.* Routledge.

[28] Roxana Moreno and Richard E Mayer. 2005. Role of guidance, reflection, and interactivity in an agent-based multimedia game. *Journal of educational psychology* 97, 1 (2005), 117.

[29] Bradford W Mott, Charles B Callaway, Luke S Zettlemoyer, Seung Y Lee, and James C Lester. 1999. Towards narrative-centered learning environments. In *Proceedings of the 1999 AAAI fall symposium on narrative intelligence*. 78–82.

[30] Multi-Dimensional. 2015. Code Spells. https://codespells.org/

[31] Igor Ostrovsky. [n.d.]. RoboZZle. http://robozzle.com/

[32] Harold F OâĂŹNeil, Gregory KWK Chung, Deirdre Kerr, Terry P Vendlinski, Rebecca E Buschang, and Richard E Mayer. 2014. Adding self-explanation prompts to an educational computer game. *Computers in Human Behavior* 30 (2014), 23–28.

[33] Neil Peirce, Owen Conlan, and Vincent Wade. 2008. Adaptive educational games: Providing non-invasive personalised learning experiences. In *2008 second IEEE international conference on digital game and intelligent toy enhanced learning*. IEEE, 28–35.

[34] Marc Prensky. 2003. Digital game-based learning. *Computers in Entertainment (CIE)* 1, 1 (2003), 21–21.

[35] Josephine M Randel, Barbara A Morris, C Douglas Wetzel, and Betty V Whitehill. 1992. The effectiveness of games for educational purposes: A review of recent research. *Simulation & gaming* 23, 3 (1992), 261–276.

[36] Jonathan Rowe, Bradford Mott, Scott McQuiggan, Jennifer Robison, Sunyoung Lee, and James Lester. 2009. Crystal island: A narrative-centered learning environment for eighth grade microbiology. In *workshop on intelligent educational games at the 14th international conference on artificial intelligence in education, Brighton, UK*. 11–20.

[37] Jennifer L Sabourin and James C Lester. 2013. Affect and engagement in Game-BasedLearning environments. *IEEE Transactions on Affective Computing* 5, 1 (2013), 45–56.

[38] Donald A Schön. 1983. The reflective practitioner: how professionals think in action.

[39] Dale H Schunk. 2012. *Learning theories an educational perspective sixth edition.* Pearson.

[40] Phoebe Sengers, Kirsten Boehner, Shay David, and Joseph'Jofish' Kaye. 2005. Reflective design. In *Proceedings of the 4th decennial conference on Critical computing: between sense and sensibility*. ACM, 49–58.

[41] Hiller A Spires, Jonathan P Rowe, Bradford W Mott, and James C Lester. 2011. Problem solving and game-based learning: Effects of middle grade students' hypothesis testing strategies on learning outcomes. *Journal of Educational Computing Research* 44, 4 (2011), 453–472.

[42] Josep Valls-Vargas, Santiago Ontanón, and Jichen Zhu. 2015. Exploring player trace segmentation for dynamic play style prediction. In *Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference*.

[43] Josep Valls-Vargas, Jichen Zhu, and Santiago Ontañón. 2017. Graph grammar-based controllable generation of puzzles for a learning game about parallel programming. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*. 1–10.

[44] Pieter Wouters, Christof Van Nimwegen, Herre Van Oostendorp, and Erik D Van Der Spek. 2013. A meta-analysis of the cognitive and motivational effects of serious games. *Journal of educational psychology* 105, 2 (2013), 249.

[45] Amri Yusoff, Richard Crowder, Lester Gilbert, and Gary Wills. 2009. A conceptual framework for serious games. In *2009 Ninth IEEE International Conference on Advanced Learning Technologies*. IEEE, 21–23.

[46] Zachtronics. 2011. SpaceChem. http://www.zachtronics.com/spacechem/

[47] Jichen Zhu, Katelyn Alderfer, Anushay Furqan, Jessica Nebolsky, Bruce Char, Brian Smith, Jennifer Villareale, and Santiago Ontañón. 2019. Programming in game space: how to represent parallel programming concepts in an educational game. In *Proceedings of the 14th International Conference on the Foundations of Digital Games*. 1–10.