Applying “Deep Gamification” Principles to Improve Quality of User-Designed Levels

Andrew Hicks, North Carolina State University
Rui Zhi, North Carolina State University
Yihuan Dong, North Carolina State University
Tiffany Barnes, North Carolina State University

Abstract: While there are many potential benefits of user-generated content for serious games, the variability of that content’s quality poses a serious problem. In our game, BOTS, players can create puzzles which are shared with other users. However, other players often find these puzzles irrelevant, unplayable, too difficult, or simply boring. This may be because content creators’ objectives when building levels differ from our own. The ‘Deep Gamification’ framework presented by Boyce et. al may help us avoid presenting players with low-quality puzzles that result in frustration, off-task behavior, and ultimately disengagement. To investigate this we have designed two new level editors for BOTS, following the Deep Gamification framework. In this paper, we discuss how the design choices made for those editors were informed by the Deep Gamification framework.

Background

Serious games and games-based learning systems have many advantages over traditional assignments. However, building educational games is costly in terms of expert time. In the Intelligent Tutoring Systems (ITS) literature it is often estimated to take between 100-300 hours of expert time to develop an hour of educational content (Murray, 1999). This estimate is insufficient for serious games for several reasons, most notably the additional expertise required to design the game and develop the game’s assets. Additionally, content must be developed to teach users how to play the game. Many serious games projects are made on small budgets by small research teams. Because of this, those serious games are relatively short in scope. If serious games were developed with enough content so that practice was not only possible but encouraged, the motivational advantages of using games or game-like systems could be amplified.

Our proposed solution to this problem is to integrate content creation (in the form of level design) as a core part of gameplay. However, our initial efforts to integrate content creation into the game met with some difficulties. Some students submitted levels in line with the game’s learning targets, while others developed levels based on entirely different objectives (Hicks, 2014). Some method of ensuring content quality is required, since providing students with low-quality exercises results in frustration, off-task behavior, and disengagement. While our initial intervention (requiring students to solve their own puzzles as a part of the authoring process) was successful at filtering out many of the negative design patterns we identified, it also somewhat reduced participation by increasing the burden on content creators, already a minority of the BOTS player base. By working within the Deep Gamification framework we believe that we will be able to design a system which accomplishes both goals, functioning as an engaging gameplay element while simultaneously improving the overall quality of user-authored levels.

Gamifying Content Creation

Gamification (or Gameful Design) is ”the use of Game Design elements in non-Game contexts” (Deterding, 2011). Though the term originated in marketing and digitial media, many of the motivations behind gamification align with those for serious games. “Traditional” gamification puts the focus on scoring and achievement systems, mimicking those in multi-player video games and social games. The assumption is that users are motivated to continue doing the gamified activity to preserve the relative standing on the scoreboard, or the richness of their collection of badges and achievements (Liu, 2011).

The issue with using this model in educational games, is that benefits only persist within the gamified system. Despite often being described as “intrinsic” motivators, these rewards are extrinsic from the task itself which suggests they may inhibit any intrinsic motivation the user may have had for the task (Deci, 1999). Additionally, adding rewards may increase participation for those “in the loop”, but it does not necessarily increase quality (Mekler, 2013). For an image-tagging task, implementing point rewards increased the number of tags submitted, but had no effect on the quality of tags submitted, while adding a framing device instead increased the quality.

Deep Gamification

In order to more completely integrate gameplay with the goals of the underlying system, Boyce et. al developed a framework called “Deep Gamification” for building engaging play experiences into educational software tools. Many of the practices outlined in Boyce’s work echo earlier work by Scott Nicholson on Meaningful Gamification (Nicholson, 2012). Where Boyce’s work diverges is in his focus on building game mechanics into existing educational software. As a result, Deep Gamification expands upon the notion of mechanical integration expressed by Nicholson, outlining many of the threats to that integration posed by common gamification practices. This framework was developed from the results of research on BeadLoom Game, in which players are required to duplicate a given image by drawing beads using functions on a Cartesian plane.

Deep Gamification was originally developed with non-games-based educational tools in mind, like the original Virtual Bead Loom tool. However, because the framework was developed in an image-creations system whose learning objectives corresponded directly to the in-game “moves” available, we can also apply its principles to content creation. This is particularly true in puzzle games where players are scored on efficiency or optimality, and where individual “moves” correspond to learning objectives. BOTS is one such game.

As outlined in Boyce’s work, Deep Gamification means:

- *The core play mechanic is precisely the learning objective*, or as near an approximation as possible. In BeadLoom Game, players solve puzzles by using iterative functions. There is no reward or resource layer between the learning activity and the core play mechanic.
- To this end, the system must *sacrifice ease of use when it conflicts with learning objectives.* Though players of BeadLoom Game often requested to be able to click the canvas to add beads, this would allow them to circumvent the learning objective of the game, and decouple the core mechanic from the learning objective.
- Where they are used, *rewards must be tightly integrated with learning outcomes.* Any activity which provides players with a reward must also be a desirable player behavior with respect to the learning objective. Arbitrarily assigned rewards ensure that players who “game the system” will be driven off-task. Care should be taken to reward improvement, not simply reward replay or re-practice (Long, 2014). The best rewards should result from demonstrating high-level understanding of the learning content.
- To this end, the system should *implement creative constraints that permit sub-optimal behavior while encouraging optimal behavior.* In BeadLoom Game, players scores are based on their ability to use iterative functions, but levels can be completed even if a player does not use them, albeit with a low rating. This allows the player to revisit problems later and improve their performance. Similarly, players are constrained in the number of operations they may use to create a level. Mastering the more complex iterative functions allows users to create more complex levels under those same constraints.
- *The system must contain both ludic (structured) and paedic (unstructured) elements.* This helps the system engage different kinds of players, as some players are not engaged or motivated by the competitive play a reward-based gamification system encourages.

An important principle for Deep Gamification is to combine learning objective with content creation. In Boyce’s BeadLoom Game, a content creation environment was created to appeal to those who disliked competitive gameplay. However, despite enjoying content creation as an activity, students were not often engaging in learning material when creating custom content (Boyce, 2011). In fact, some users actively avoided learning objectives when creating custom designs, building images pixel-by-pixel even though they had used the iterative tools previously. To address this, they reworked their level editor, applying these same principles to the content creation tool as to the original educational tool. This approach is what we will apply to our game, BOTS.

Overview of BOTS

BOTS (bots.game2learn.com) is a puzzle game designed to teach fundamental ideas of programming and problem-solving to novice computer users. BOTS was inspired by games like *LightBot* and *RoboRally*, as well as the syntax of *Scratch* and *Snap* (Garfield, 1994; Armor Games, 2010). In BOTS, players take on the role of programmers writing code to navigate a simple robot around a grid-based 3D environment. The goal of each puzzle is to press several switches within the environment, which can be done by placing an object (or the robot itself) on top of them. Within each puzzle, players’ scores depend on the number of commands used, with lower scores being preferable. In addition, each puzzle limits the maximum number of commands, as well as the number of times each command can be used (see Figure 1). For example, in the tutorial levels, a user may only use the ”Move Forward” instruction 10 times. Therefore, if a player wants to make the robot walk down a long hallway, it will be more efficient to use a loop to repeat a single ”Move Forward” instruction, rather than to simply use several ”Move Forward” instructions one after the other. These constraints, based on the Deep Gamification framework, are meant to encourage players to optimize their solutions by practicing loops and functions.

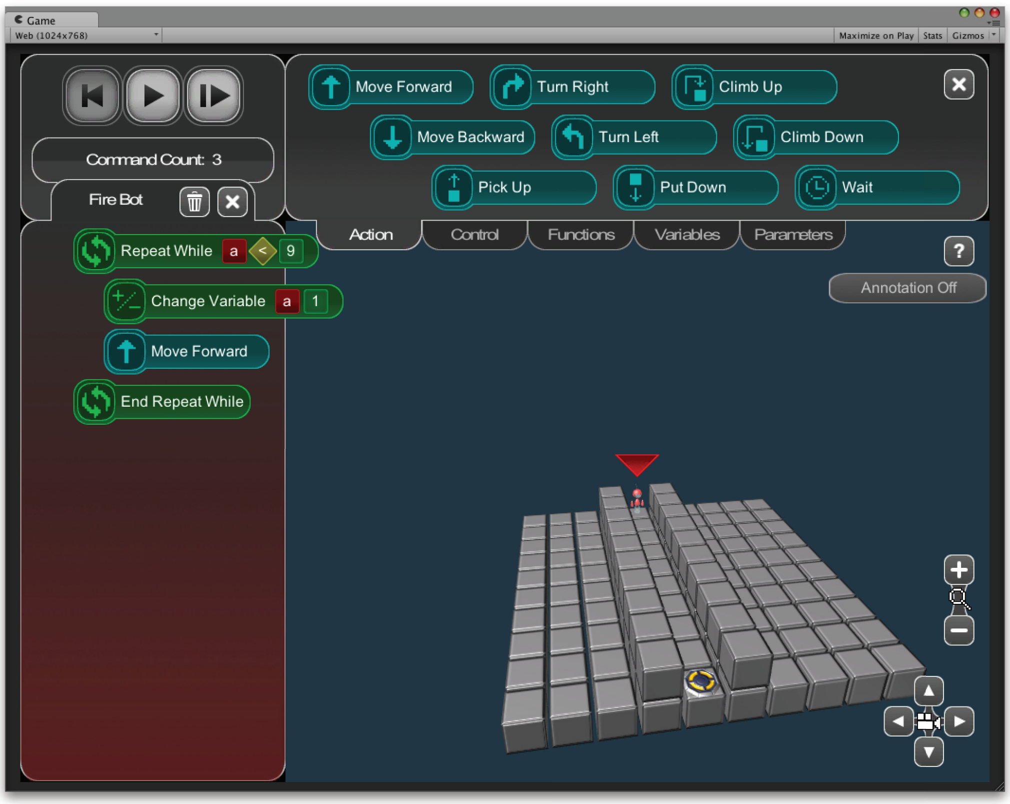


Figure 1: An early level in BOTS, demonstrating how use of loops can simplify repetitive tasks.

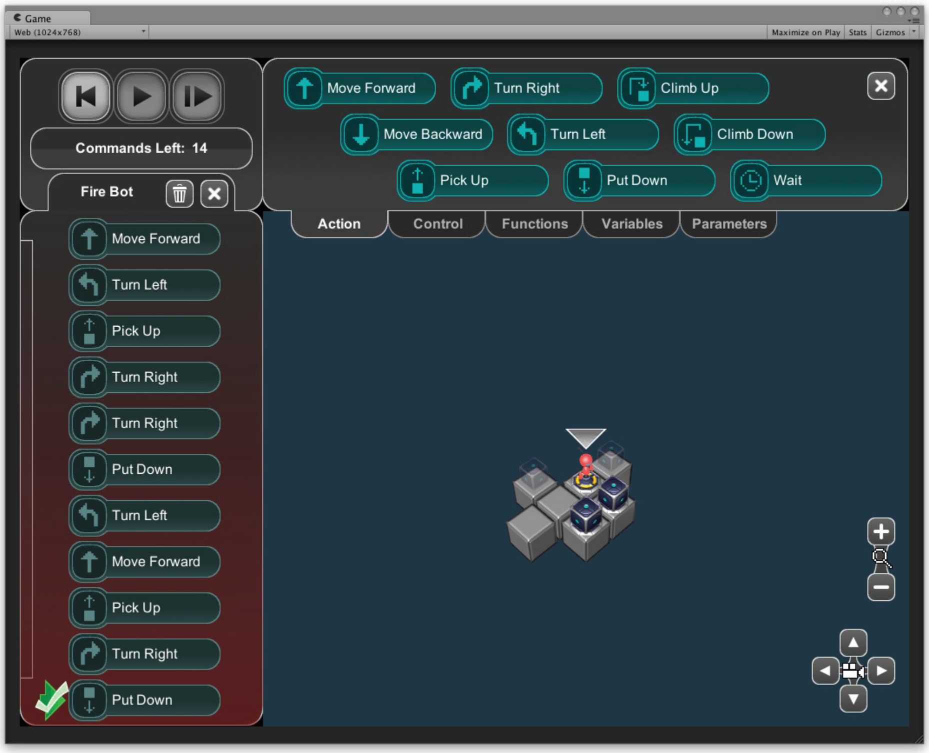
Application of Deep Gamification to BOTS

In the free-form version of the level editor, players are free to drag-and-drop elements into a blank level which they must solve after they submit it. While players often created content of various negative patterns, requiring them to also provide a valid solution after submitting was successful at eliminating many of the negative patterns of content. We theorized that these negative patterns occurred when users’ objectives during content creation were very different from our own. Where we are primarily concerned with the complexity and content of the solutions, players often created structures or patterns that were more visually interesting, but did not afford desirable solutions.

Having previously shown that some levels/problems created by users are of sufficient quality to be used as practice exercises (Hicks, 2014), our next step is to make further improvements to the content authoring tools. We decided to add objectives and constraints to the level editor, in order to help align players’ goals with our own, and thus increase the overall quality of submitted content. To this end we designed two new versions of the game’s level editor, with two different types of constraints.

Both level editors adhere to the Deep Gamification framework as outlined above, but one draws additional inspiration from structured problem-posing activities used in mathematics education. We propose to evaluate level editors with two different forms of constraint added. The *Programming Editor*, where the length (in lines of code) of the solution is constrained, similarly to the Point Value Showcase in BeadLoom Game. Second, the *Block-Based Editor*, where the construction of the level itself is constrained by providing authors with a limited selection of ‘‘building blocks’’ for which partial solutions are provided, and requiring users to improved upon the final solution before submitting the level.

The design of the Programming Editor (shown in Figure 2) is based directly upon the level editor in BeadLoom Game (Boyce, 2012). While using this editor, players are able to create a level by programming the path the robot will take. Players are constrained to using a limited number of instructions. This is analogous to the level creation tools in BeadLoom Game where players created levels for various “showcases” under similar constraints. This type of constraint has been shown to be effective for encouraging players to perform more complex operations in order to generate larger more interesting levels under the constraints. One challenge with this approach is that since simple solutions are still permitted, and most programs are syntactically correct, users who are experimenting with the level creation interface with no goal in mind may be able to create levels that they themselves do not understand.



```
Move Forward
Turn Left
Pick Up
Turn Right
Turn Right
Put Down
Turn Left
Move Forward
Pick Up
Turn Right
Put Down
```

Figure 2: Screenshot of the Programming Editor in BOTS. Developing complex levels is easier and less time-consuming than with the previous drag-and-drop editor.

To summarize:

- *The core play mechanic is precisely the learning objective*, since players use the programming interface to create levels.
- The system does *sacrifice ease of use when it conflicts with learning objectives*, since pointing-and-clicking to create a new puzzle would be simpler but would decouple content creation and learning objective..
- *Rewards are tightly integrated with learning outcomes* since the same scoring system used for gameplay is used for creation, with lower numbers of lines of code representing a better outcome.
- The system *implements creative constraints that permit sub-optimal behavior while encouraging optimal behavior.* Without using loops and functions, only simple layouts are possible. Using loops permits the construction of complex levels.

Alternatively, the Block-Based Editor constrains level creation by providing known meaningful chunks to authors in the form of ‘’building blocks.’’ This is inspired by problem-posing activities as presented in systems like MONSAKUN and AnimalWatch, in which players are asked to build a problem using data and problem pieces provided by experts (Hirashima, 2007; Birch, 2008). In this version of the level editor, players will be asked to create a level only using pre-constructed chunks of levels (Figure 3). These ‘’building blocks’’ will be specific structures which correspond to opportunities to use loops, functions, or variables. Some examples of this are shown in Figure 4.

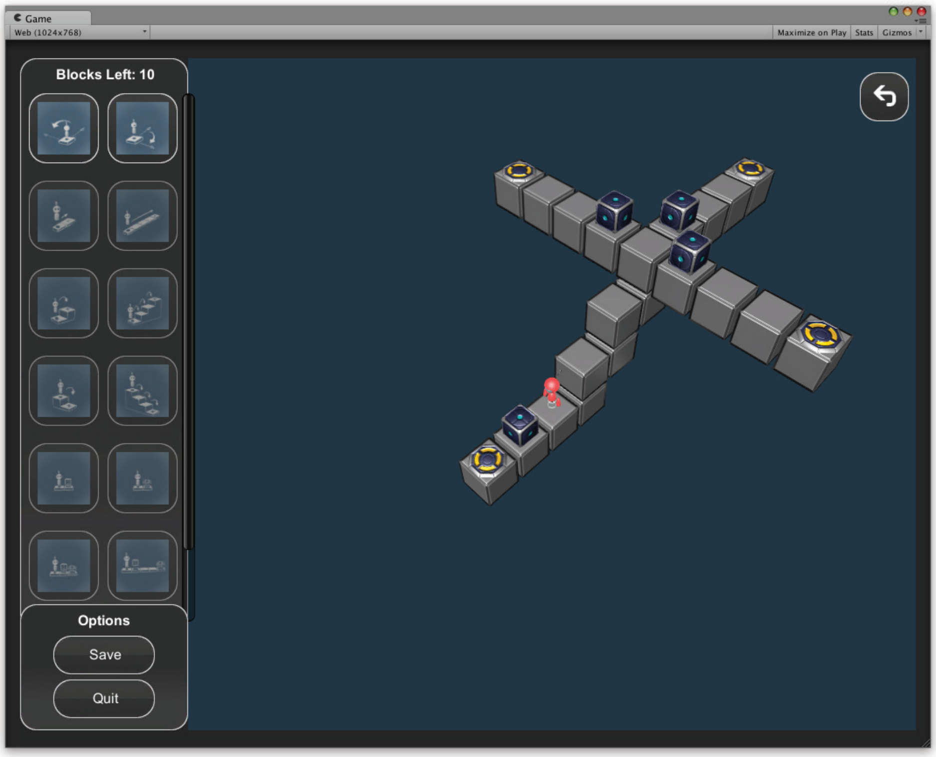


Figure 3: Screenshot of the "Building Block" interface, with a "palette" of building blocks. Developing levels which contain opportunities for use of loops and functions is simplified, since the building blocks are components of such levels.

As players build a level by selecting from these blocks, each block will be appended to the last, with the new block's starting position overlaid on the previous block's ending position. At the same time, a composite solution made up of the simple solutions for those blocks will be built. Players are constrained to a limited number of building blocks, which As the final step of submitting a level, the author must provide a solution, which will be compared against the composite solution for determining score. This will encourage authors to look for opportunities to optimize while building the level, either within the building blocks, or by using the same block multiple times.

To summarize:

- *The core play mechanic is precisely the learning objective*. Since players must optimize the composite solution to submit their level, they must recognize opportunities for optimization while building the level. This is a more abstracted version of the learning objective than above.
- The system does *sacrifice ease of use when it conflicts with learning objectives,* since players are constrained to working with pre-determined elements, and cannot create the level exactly as they would like.
- *Rewards are tightly integrated with learning outcomes* in the same way as with the Programming Editor, since the same scoring system used for gameplay is used for creation, with lower numbers of lines of code representing a better outcome.
- The system *implements creative constraints that permit sub-optimal behavior while encouraging optimal behavior*. While the building blocks do contain opportunities for optimization, the author is not required to submit the most optimal solution. Reducing the size of the program by a single line is sufficient, but the level will be listed higher if a more optimal solution exists.

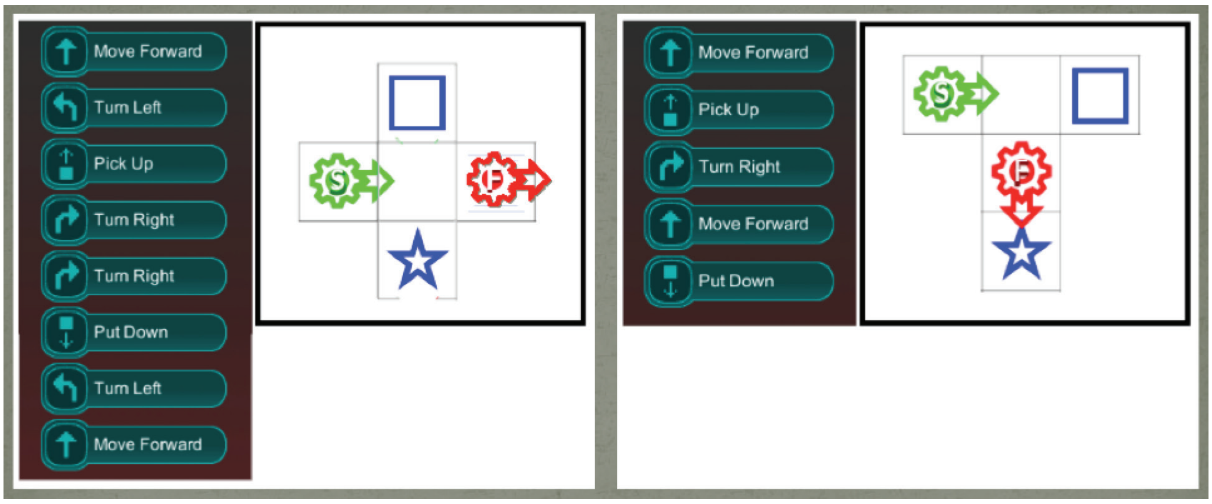


Figure 4: Example building blocks, derived from repeated patterns in previously created levels.

We hypothesize that this will lead to better levels because it explicitly promotes the inclusion of these patterns, which will lead to opportunities for players to use more complex programming constructs like loops and functions. We also believe that this will encourage students to think about optimizing the solution to the level while they are building it. One potential challenge with this approach is that students may find these constraints too restrictive, which might reduce engagement for creatively-oriented players.

Conclusion

While initially developed for use in gamifying non-playful educational tools like Intelligent Tutoring Systems, the framework of Deep Gamification can also be applied to games with elements that in themselves are not playful, such as puzzle design. We have applied this framework to develop two additional modes of content creation for our game, BOTS, and in future work we will compare levels created by players under both approaches to gain additional insight into how integrating these game mechanics with content creation affects players' participation in and engagement with the content creation activity, as well as the quality (in terms of complexity and learning opportunities) of the content created.

Acknowledgments

Thanks to the additional developers who have worked on this project or helped with our outreach activities so far, including Aaron Quidley, Veronica Catete, Alex Milliken, Trevor Brennan, Barry Peddycord, Vincent Bugica, Victoria Cooper, Dustin Culler, Shaun Pickford, Antoine Campbell, and Javier Olaya. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. 0900860 and Grant No. 1252376.

References

Armor Games. (2010). Light-Bot 2.0 [Flash game]. http://www.armorgames.com

Boyce, A. (2014). Deep Gamification: Combining Game-based and Play-based Methods.

Boyce, A., Doran, K., Campbell, A., Pickford, S., Culler, D., & Barnes, T. (2011). BeadLoom Game: adding competitive, user generated, and social features to increase motivation. In *Proceedings of the 6th International Conference on Foundations of Digital Games* (pp. 139-146). ACM.

Boyce, A., Campbell, A., Pickford, S., Culler, D., & Barnes, T. (2012). Maximizing learning and guiding behavior in free play user generated content environments. In *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education* (pp. 10-15). ACM.

Boyce, A., Doran, K., Campbell, A., Pickford, S., Culler, D., & Barnes, T. (2011). Social user generated content's effect on creativity in educational games. In *Proceedings of the 8th ACM conference on Creativity and cognition* (pp. 317-318). ACM.

Beal, C. R., & Cohen, P. R. (2012). Teach Ourselves: Technology to Support Problem Posing in the STEM Classroom. *Creative Education*, *3*(04), 513.

Birch, M., & Beal, C. R. (2008, May). Problem Posing in AnimalWatch: An Interactive System for Student-Authored Content. In *FLAIRS Conference* (pp. 397-402).

Deci, E. L., Koestner, R., & Ryan, R. M. (1999). A meta-analytic
review of experiments examining the effects of extrinsic rewards on intrinsic motivation.*Psychological bulletin*, *125*(6), 627.

Deterding, S., Khaled, R., Nacke, L. E., & Dixon, D. (2011). Gamification: Toward a definition. In *CHI 2011 Gamification Workshop Proceedings* (pp. 12-15).

Garfield, R. (1994). *RoboRally*. Seattle, WA. Wizards of the Coast.

Hashimoto, Y. (1987). Classroom practice of problem solving in Japanese elementary schools. In *Proceedings of the US-Japan seminar on mathematical problem solving. Columbus, OH: ERIC/SMEAC Clearinghouse (ED 304 315)*(pp. 94-119).

Hicks, A. (2012). Creation, evaluation, and presentation of user-generated content in community game-based tutors. In *Proceedings of the International Conference on the Foundations of Digital Games* (pp. 276-278). ACM.

Hicks, A., Cateté, V., & Barnes, T. (2014). Part of the Game: Changing Level Creation to Identify and Filter Low Quality User-Generated Levels. In *Proceedings of the International Conference on the Foundations of Digital Games*

Hirashima, T., & Kurayama, M. (2011). Learning by problem-posing for reverse-thinking problems. In *Artificial Intelligence in Education* (pp. 123-130). Springer Berlin Heidelberg.

Hirashima, T., Yokoyama, T., Okamoto, M., & Takeuchi, A. (2007). Learning by problem-posing as sentence-integration and experimental use. In *AIED* (Vol. 2007, pp. 254-261).

Liu, Y., Alexandrova, T., & Nakajima, T. (2011). Gamifying intelligent environments. In *Proceedings of the 2011 international ACM workshop on Ubiquitous meta user interfaces* (pp. 7-12). ACM.

Long, Y., & Aleven, V. (2014). Gamification of Joint Student/System Control over Problem Selection in a Linear Equation Tutor. In *Intelligent Tutoring Systems* (pp. 378-387). Springer International Publishing.

Mekler, E. D., Brühlmann, F., Opwis, K., & Tuch, A. N. (2013). Disassembling gamification: the effects of points and meaning on user motivation and performance. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems* (pp. 1137-1142). ACM.

Nicholson, S. (2012). A user-centered theoretical framework for meaningful gamification. *Games+ Learning+Society*, *8*(1).