Generating Game Levels to Develop Computer Science Competencies in Game-Based Learning Environments

Kyungjin Park[1], Bradford Mott[1], Wookhee Min[1], Eric Wiebe[1], Kristy Elizabeth Boyer[2], and James Lester[1]

[1] North Carolina State University, Raleigh, NC 27606, USA
{kpark8, bwmott, wmin, wiebe, lester}@ncsu.edu
[2] University of Florida, Gainesville, FL 32601, USA
{keboyer}@ufl.edu

Abstract. Game-based learning environments hold significant potential for supporting K-12 computer science (CS) education by providing CS learning experiences embedded within engaging virtual worlds. However, many game-based learning environments do not adaptively support individual students based on their specific knowledge and skills. Often, this is because creating game levels is highly labor-intensive, which limits the number of levels created to support student learning. Procedural content generation (PCG) is a promising direction for addressing this challenge by dynamically creating game levels that address specific student needs without requiring extensive development effort. In this paper, we investigate a PCG framework driven by answer set programming (ASP), a variant of logic programming that utilizes well-formed logical rules to express constraints for valid game levels. We demonstrate how variations in CS learning objectives and game-playing skills can be incorporated into ASP-based rules to generate learner-adaptive levels in a middle-grades CS game-based learning environment. Evaluations of the generated levels suggest that the ASP-based level generator not only reliably generates desired CS educational game levels but also synthesizes a large set of diverse game levels. The findings suggest that the ASP-based PCG approach has considerable promise for creating highly engaging and adaptive game-based learning experiences for K-12 CS education.

Keywords: K-12 computer science education, game-based learning, procedural content generation, answer set programming.

1 Introduction

Recent years have seen growing interest in game-based learning environments [1-4], which engage students in situated problem-solving challenges within rich virtual worlds [5]. In parallel, there is a growing recognition that computer science (CS) is a fundamental skill required by many career paths, which has intensified the need to develop K-12 students' CS competencies [6-9] and highlighted the potential of game-based learning environments to support CS education [10-12]. However, the conventional approach of utilizing a linear sequence of game levels is fundamentally non-adaptive and may not effectively address the needs of different students based on their level of concept and skill mastery. This lack of adaptivity may result in undesirable learning experi-

ences (e.g., students adopting a trial-and-error approach without mastering concepts because a game-based learning environment is too difficult). Likewise, students have different levels of game-playing skills, which can affect their learning experiences [13]. Thus, adaptively generating challenges tailored to individual students' knowledge and game-playing skill is crucial for supporting mastery learning and engagement in game-based learning by addressing limitations with "one-size-fits-all" approaches.

Procedural content generation (PCG) automatically generates game content using a range of algorithms that require limited human intervention [14]. In contrast to problem generation in intelligent tutoring systems, in which problems are generated using templates [15, 16], PCG explores the generation of game objects and their layout that collectively constitute a game level. However, level generation in game-based learning environments is challenging for PCG because game levels must exercise the desired learning objectives for individual students as well as target an appropriate level of difficulty for students based on their game-playing skill.

This paper presents a novel approach to generating game levels for game-based learning environments. Our work is the first to introduce a PCG framework that dynamically generates game levels to develop individual students' CS competencies using answer set programming (ASP) [17]. We evaluate our framework with respect to the diversity of generated game levels and the presence of the CS learning objectives as well as the game-playing skill specified as input for each generated level in the context of a game-based learning environment for middle school CS education.

2 ASP-Based Level Generation in ENGAGE

ENGAGE is a game-based learning environment for middle school CS education, the curriculum of which is guided by the K-12 CS Framework [18]. In ENGAGE, students play the role of a protagonist who is sent to an undersea research station, where a rogue villain has severed communication with the facility. In this work, we focus on generating levels for a specific type of challenge shown in Fig. 1a which requires students in the game to connect their wrist computer with a quadcopter device using a pairing point, and program the quadcopter to navigate across a water-filled area while avoiding obstacles. Fig. 1b shows a top-down view of the room, which serves as the basis of all the generated levels in this work.

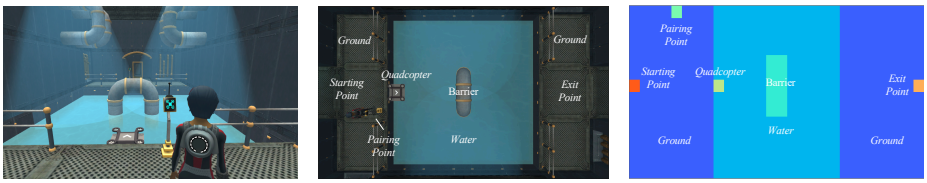

Fig. 1. (a) In-game 3D view of the level, (b) top-down view, and (c) 2D tile-based representation.

Generated levels incorporate four key learning concepts, *Loop*, *Conditional*, *Sequence* (i.e., requiring minimum of two controls in an unnested structure), and *Nested Control* (i.e., requiring at least one nested control structure), based on the core computer science concepts delineated in the K-12 CS Framework [18], and three game-playing

skills (*Low*, *Medium*, *High*) based on the required number of jumps and the width of the path the student's in-game avatar must navigate. To visualize the generated levels, we use a 2D tile-based level representation, as depicted in Fig. 1c.

Answer set programming (ASP) is a declarative programming paradigm, which has its roots in logic programming. In ASP-based PCG, a set of basic requirements and constraints needed for content generation is represented in logical terms (i.e., rules and ground facts) [19]. Then a solver (e.g., Clingo [20]) produces all configurations of content (e.g., game levels) that satisfy the specified constraints. ASP utilizes two constructs: 1) *Choice Rules* to enable non-determinism in choosing ground facts, and 2) *Integrity Constraints* which explicitly define what must not be true in the logical world. Table 1 shows the specific constraints for the four CS learning objectives as well as the three different rulesets for the game-playing skill variations we are considering in ENGAGE.

Table 1. Level category-specific *Choice Rules* and *Integrity Constraints*.

Category	Choice Rules	Integrity Constraints
Loop	The number of repetitive parts.	There exists only one path that goes through the repetitive pattern.
Conditional	Position of the conditional tile.	There exists only one path that passes through the conditional tile.
Sequence	Conditional tile exists either at the start of the loop or at the end of loop.	There exists only one path that requires a sequence programming.
Nested Control	Conditional tile exists anywhere within a repetitive pattern.	There exists only one path that requires nested control programming.
Game Skills	Positions where a jump is required Lower the number of connected ground tiles towards *High* level.	The character can jump up to one tile. The character can move diagonally. The character cannot jump diagonally.

3 Evaluation

Quantitative Evaluation. We measure the diversity among 100 levels created by the ASP-based level generator using the Clingo [20] solver for each of the 12 categories (four learning concepts combined with three game-playing skills) using a coordinate-based distance metric presented in previous works [21, 22]. The average diversity values of the ASP-generated levels within each category are shown in Table 2. A diversity of 0 indicates that every matched pair of tile types between two levels is identical, while 1 indicates there are no tile types in common across the levels. The average diversity score across all 12 categories is 0.290, which indicates that 29% of tiles (i.e., 113 tiles out of 392 tiles) different between any pair of randomly chosen levels on average. This demonstrates that our model generates levels different to a certain degree consistently. While most categories achieved high diversity scores, *Low* game-playing skill levels across all CS concepts show comparatively lower scores because fewer variations are available within the walkable ground area in these levels.

Table 2. Diversity of 100 levels generated for each of the 12 categories

Loop			***Conditional***			***Sequence***			***Nested Control***			**Avg.**
Low	***Med.***	***High***	***Low***	***Med.***	***High***	***Low***	***Med.***	***High***	***Low***	***Med.***	***High***	
0.132	0.327	0.299	0.234	0.244	0.369	0.135	0.248	0.307	0.135	0.248	0.307	0.290

Qualitative Evaluation. Two domain experts evaluated each level with respect to the presence of the CS learning objectives as well as the game-playing skill required for the level. The evaluators rated each level with game-playing skill (*Low*: 1, *Medium*: 2, *High*: 3) and one binary value for each of the four CS concepts, where 1 indicates the desired concept is present in the level, while 0 is not. The values reported in Table 3 are the averages of the two evaluators' ratings for 100 generated levels. Results for presence of CS concepts suggest that *Sequence, Loop,* and *Conditional* exhibit complete agreements between the human raters, while comparably less agreement occurs for the *Nested Control.* This phenomenon can be explained because some levels have a conditional barrier at the front or end of a path with a repetitive pattern that does not necessarily require use of nested blocks (e.g., it can be solved with a loop followed by a conditional block). Also, we found that there is a small degree of disagreement between *Medium* and *High* game-playing skill levels, while *Low* skill levels were consistently viewed as *Low*.

Table 3. Average human-evaluated presence of CS concepts and game-playing skills (GS).

ASP	*Loop*			*Conditional*			*Sequence*			*Nested Control*		
	***Low* (1)**	***Med.* (2)**	***High* (3)**	***Low* (1)**	***Med.* (2)**	***High* (3)**	***Low* (1)**	***Med.* (2)**	***High* (3)**	***Low* (1)**	***Med.* (2)**	***High* (3)**
CS	1	1	1	1	1	1	1	1	1	0.55	0.8	0.65
GS	1.05	2.55	2.85	1.2	2.1	2.65	1	2.4	2.95	1	1.95	2.75

4 Conclusion

Game-based learning environments show significant promise for creating engaging learning experiences for students. However, manually crafting a large number of game levels, which is typically required to adaptively support students' mastery learning, is labor-intensive. In this work, we presented an ASP-based PCG framework that automatically synthesizes game levels, and we investigated its generation capabilities for a middle-grade CS game-based learning environment. Evaluation results suggest that the ASP-based level generation framework creates diverse levels, while dynamically synthesizing levels that capture both the learning and game-playing skill-focused specifications. Together, our framework shows significant potential for offering adaptive CS learning experiences with enhanced replayability. In the future, it will be important to investigate robust student modeling techniques to inform the decision-making of the PCG framework to provide student competency-adaptive levels and effectiveness of personalized levels in terms of developing students' CS competencies.

Acknowledgements

This research was supported by the National Science Foundation under Grant DRL-1640141. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

1. Clark, D. B., Tanner-Smith, E. E., Killingsworth, S. S.: Digital games, design, and learning: A systematic review and meta-analysis. Review of Educational Research, 86(1), 79-122 (2016).
2. Easterday, M. W., Aleven, V., Scheines, R., Carver, S. M.: Using tutors to improve educational games. In: Proceedings of the International Conference on Artificial Intelligence in Education, pp. 63-71. Springer, Berlin, Heidelberg (2011).
3. Nguyen, H., Harpstead, E., Wang, Y., McLaren, B. M.: Student agency and game-based learning: A study comparing low and high agency. In: Proceedings of the International Conference on Artificial Intelligence in Education, pp. 338-351. Springer, Cham (2018).
4. Jackson, G. T., Dempsey, K. B., McNamara, D. S.: Short and long term benefits of enjoyment and learning within a serious game. In: Proceedings of the International Conference on Artificial Intelligence in Education, pp. 139-146. Springer, Berlin, Heidelberg (2011).
5. Spires, H. A., Rowe, J. P., Mott, B. W., Lester, J. C.: Problem solving and game-based learning: Effects of middle grade students' hypothesis testing strategies on learning outcomes. Journal of Educational Computing Research, 44(4), 453-472 (2011).
6. Grover, S., Basu, S., Schank, P.: What we can learn about student learning from open-ended programming projects in middle school computer science. In: Proceedings of the 49th ACM Technical Symposium on Computer Science Education, pp. 999-1004. ACM (2018).
7. Nouri, J., Zhang, L., Mannila, L., Norén, E.: Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. Education Inquiry, 11(1), 1-17 (2020).
8. Rich, K. M., Strickland, C., Binkowski, T. A., Moran, C., Franklin, D.: K-8 learning trajectories derived from research literature: Sequence, repetition, conditionals. In: Proceedings of the 2017 ACM Conference on International Computing Education Research, pp. 182-190. ACM (2017).
9. Weintrop, D., Hansen, A., Harlow, D., Franklin, D.: Bringing computer science into elementary school classrooms. American Educational Research Association (2018).
10. Hicks, A., Dong, Y., Zhi, R., Cateté, V., Barnes, T.: BOTS: Selecting next-steps from player traces in a puzzle game. In: Proceedings of the Second International Workshop on Graph-Based Educational Data Mining (2015).
11. Bauer, A., Butler, E., Popović, Z.: Dragon architect: Open design problems for guided learning in a creative computational thinking sandbox game. In: Proceedings of the 12th International Conference on the Foundations of Digital Games, pp. 1-6. ACM (2017).
12. Min, W., Frankosky, M. H., Mott, B.W., Wiebe, E., Boyer, K.E., Lester, J.C.: Inducing stealth assessors from game interaction data. In: Proceedings of the International Conference on Artificial Intelligence in Education, pp. 212-223. Springer, Cham (2017).
13. Rowe, J. P., Shores, L. R., Mott, B. W., Lester, J. C.: Integrating learning, problem solving, and engagement in narrative-centered learning environments. International Journal of Artificial Intelligence in Education, 21(1-2), 115-133 (2011).
14. Togelius, J., Kastbjerg, E., Schedl, D., Yannakakis, G. N.: What is procedural content generation? Mario on the borderline. In: Proceedings of the 2nd International Workshop on Procedural Content Generation in Games. ACM (2011).
15. Singh, R., Gulwani, S., Rajamani, S.: Automatically generating algebra problems. In: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (2012).
16. Gierl, M. J., Lai, H., & Turner, S. R.: Using automatic item generation to create multiple-choice test items. Medical education, 46(8), 757-765 (2012).

17. Smith, A. M., Mateas, M.: Answer set programming for procedural content generation: A design space approach. IEEE Transactions on Computational Intelligence and AI in Games, 3(3), 187-200 (2011).
18. K-12 Computer Science Framework. https://k12cs.org/ (2016).
19. Sterling, L., Shapiro, E. Y.: The art of Prolog: Advanced programming techniques. MIT press (1994).
20. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Clingo = ASP+ control: Preliminary report. arXiv preprint arXiv:1405.3694 (2014).
21. Park, K., Mott, B. W., Min, W., Boyer, K. E., Wiebe, E. N., Lester, J. C.: Generating educational game levels with multistep deep convolutional generative adversarial networks. In: Proceedings of the 2019 IEEE Conference on Games (CoG), pp. 345-352. IEEE (2019).
22. Liapis, A., Yannakakis, G. N., Togelius, J.: Enhancements to constrained novelty search: Two-population novelty search for generating game content. In: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, pp. 343-350. ACM (2013).