7 Conclusion

In this work, we introduced and evaluated a novel approach for content generation for games, which we call solution-down procedural content generation via reinforcement learning (SD-PCGRL), by using it to generate levels for the hyper-casual puzzle game Longcat. We also generated not only solvable levels, but also made them engaging and unique. We chose to define engagement around challenging the player, meaning that we aimed to find a generator that can generate levels of a variety of difficulties. We also compared SD-PCGRL to three other generation methods: an adapted version of the original PCGRL method, a random generator based on the existing one in the Longcat application, and a random solution-down generator.

For the evaluation and training, we needed a way to determine the difficulty of Longcat levels, so we also analyzed Longcat levels and, in combination with real user data, determined what aspects of a Longcat level are important for predicting difficulty.

With this approach we formulated the following research questions and found subsequent answers.

1. What attributes of Longcat levels are relevant for predicting difficulty?

   From our analysis phase, we formulated a graph representation for Longcat levels. Variables of these graphs were then used to attempt to predict difficulty. We found that Number of indeterminate branches, Number of solutions, and Number of solution branches were the most useful variables for prediction. Shortest solution length and Number of indeterminate states were also shown to be strong predictors, but they were left unused since they were exploitable by the generator.

2. What percentage of Longcat levels generated with PCGRL and Puzzle-Up are solvable?

   The PCGRL model, based on the turtle method from the original article, was unable to create any solvable levels. The puzzle-up random generator based on the one in the Longcat application generated solvable levels 5.535% of the time. Both these generators use a type of process called puzzle-up, which, from findings, seems to be very ill-suited for generating Longcat levels. The other two generators use a type of process called solution-down that guarantees solvability.

3. What are the differences in difficulty distribution for Longcat levels generated with SD-PCGRL compared to those generated with a random generator?

SD-PCGRL has a more spread-out distribution at a lower non-zero temperature, than the ones generated with the random generators. This means it generates fewer levels of trivial and easy difficulty and more of a challenging difficulty. Its most challenging levels are also more difficult than the random generator's most demanding levels.

4. What percentages of Longcat levels generated with SD-PCGRL are duplicates, and how does it compare with a random generator?

   The puzzle-up random generator gave 68.56% of generated levels as duplicates. The solution-down random generator gave 9.68% of generated levels as duplicates. SD-PCGRL was shown to have a similar number of duplicates as the solution-down random generator at higher temperatures, but the number of duplicates increases as the temperature gets lower. At a temperature of 0.3, which is the best temperature for difficulty distribution, it gave 41.0% of generated levels as duplicates.

5. What percentages of Longcat levels generated with SD-PCGRL have a non-trivial difficulty, and how does it compare with a random generator?

   Of the levels generated with the SD-PCGRL generator, 28.64% of them were solvable, unique, and non-trivial, while only 8.24% were for the solution-down random generator.

From this, we found that the SD-PCGRL was more useful for generating engaging puzzle levels for Longcat than the other generators. We believe this approach of using solution-down generation, difficulty predictors, and reinforcement learning could be used to develop generators for different games. However, this would require a lot of specialized work from a designer, which might be possible for some games. Ultimately, the methods described show promise but must be tested with other games to verify their usefulness.