Procedural Level Design using an Interactive Cellular Automata Genetic Algorithm[*]

Chad Adams
University of Nevada, Reno
Reno, NV 89557
adams.chad7@gmail.com

Hirav Parekh
University of Nevada, Reno
Reno, NV 89557
hirav.parekh@gmail.com

Sushil J. Louis
University of Nevada, Reno
Reno, NV 89557
sushil@cse.unr.edu

ABSTRACT

In this paper we propose an interactive genetic algorithm to evolve maze levels for computer games. We represent a maze with a cellular automaton and the genetic algorithm evolves the cellular automata rules applied to a starting maze level state. Users then rate the fitness of a subset of the generated population using an image of the top-down view of the maze. After ten generations, users then play through the best evolved maze within a maze runner type game using a first person perspective. User ratings show that our IGA was able to evolve highly rated mazes.

CCS CONCEPTS

•Theory of computation → Evolutionary algorithms; •Applied computing → *Computer games;*

KEYWORDS

Evolutionary Computing, Genetic Algorithm, Procedural Content

ACM Reference format:
Chad Adams, Hirav Parekh, and Sushil J. Louis. 2017. Procedural Level Design using an Interactive Cellular Automata Genetic Algorithm. In *Proceedings of GECCO '17 Companion, Berlin, Germany, July 15-19, 2017,* 2 pages.
DOI: http://dx.doi.org/10.1145/3067695.3075614

1 INTRODUCTION

This paper explores an interactive approach utilizing cellular automata. We show that human interaction can be used to guide an evolutionary search of procedural level generation for mazes.

Maze running games are a type of game where the player speeds through a maze to find an objective in the maze. The design of the maze makes these games fun to play and traditionally, expensive game level designers handcrafted interesting and fun mazes for maze running games. We use an Interactive Genetic Algorithm (IGA) to evolve mazes for maze running games.

In general, procedural content generation refers to the concept of automated game level (or map) design. There are several mainstream games that create their content procedurally (instead of using a level designer) such as the famous Diablo game series [1].

[*]This work is partially supported by the Office of Naval Research under Grant N00014-15-1-2015.

Compton and Mateas split levels into pieces, algorithmically generate the design on each piece and then combine the pieces together to generate a complete level for their platformer game [4]. Smith et. al. designed a level generator for 2-D platformer game that uses a model of player action-rhythm [10] to generate levels of appropriate difficulty. In Johnson et. al. Cellular Automata were used to generate infinite cave level maps [7]. Procedural content generation has also been used to dynamically adjust a level's difficulty [6] and to study the effects of level design on player behavior [9].

Closer to our work, Marks and Hom [5] evolved a set of board game rules using a GA to make the game equally hard to win for either side. Cardamone et al. use a GA to evolve maps to maximize a fitness function based on the player's average fighting time [3].

L. R. Smith's work on evolving Fourier series "biomorphs" [11] using an interactive genetic algorithm for procedural content generation can be considered early work on generating player characters. Later, IGA's were applied to track generation for the TORCs racing simulator [2] game where users assigned fitness to tracks. This work, to the best our knowledge, is the closest to the work done in this paper as we use a similar approach for users assigning fitness. Our work differs in using cellular automata to construct mazes instead of the control point method used in Cardamone's work [2].

2 METHODOLOGY

We used Unity3D, along with the code for cave generation from Sebastian Laguefis Cave Generation Tutorial [8]. The chromosomes used contain the cellular automata rules and starting map state. Using region merging afterwards we were able to guarantee a complete maze level [8]. Because IGAs suffer from user fatigue [2], we used a small population of size sixteen and only presented the user with the first nine non-duplicate mazes from the population.

2.1 Interactive Genetic Algorithm

We used fitness proportional selection, two point crossover, and point mutation. Our chromosome is represented as a one dimensional array of bits, the first four hundred representing the starting map state for our 20×20 maze level, and the remaining eighteen representing the cellular automata rules applied to the map. We used two point crossover, but one point was constrained to the starting map state section and the other crossover point was constrained to the cellular automata rules section. We use the canonical GA rather than a more elitist method as we want to explore the adaptive landscape and retain some diversity in our population.

We have eighteen bits in our chromosome for encoding the rules of a cellular automata. Since a cell can have a maximum of nine neighbors including itself, we split the rules into two parts. The first nine bits specify the bit mask for changing cell state to be filled, the last nine bits specify the bit mask for changing cell state to blank (or unfilled).

2.2 Fitness Evaluation

We present only a subset of nine mazes from the population for user evaluation using sliding bars below each maze and a top down view. The seven mazes that are not shown are given the fitness of the maze that has the shortest hamming distance to the maze.

3 RESULTS AND DISCUSSION

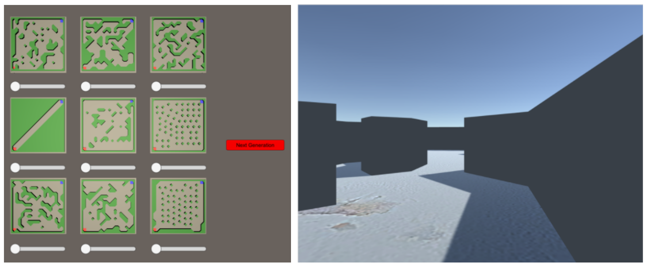

Figure 1: Left: Evolution mode. Right: Maze running mode

We ran our IGA with seven different users for ten generations per user. After the user finished, we asked the user to play the highest ranked level from the initial population, the highest ranked level from the fifth generation, and the highest ranked level from the last generation. "Playing" a maze, meant that the user saw the game from a first person perspective as shown in Figure 1 on the right. Users then rated the play-through of the level on a Likert scale of one to five. The highest fitness mazes from the initial population received an average rating of 2.75, with the middle and final generation best mazes getting better at 3.25 and 3.75 respectively. We can see the improvement over time in Figure 2.

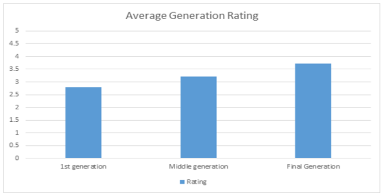

Figure 2: Average scale rating for each sample generation

We did encounter users that rated the final generation levels as less fun than middle generation levels because the final generation levels were more difficult. We conjecture that perhaps more difficult levels are more fun, but only up to a point.

3.1 Level Metrics and Behaviour

After collecting the evolved mazes and their Likert ratings we then evaluated the mazes on more objective criteria. We considered the number of dead ends, non-looping solution paths, the number of turns on the shortest solution path, and the number of unconnected wall sections, not including the outer border wall. Figure 3 shows

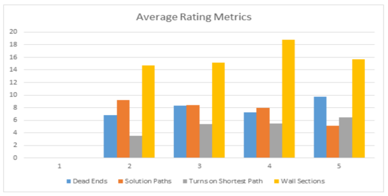

Figure 3: Average metrics for each rating

how these metrics map to Likert scale values for the mazes that were evolved. The number of solution paths and number of turns on the shortest path follow a trend where the higher rated a level the fewer solution paths exist and with more turns on those paths. The number of dead ends and wall sections go up and down. We believe this was because we had a larger number of trivial dead ends on 3 rated mazes compared to 4 rated mazes.

4 CONCLUSION AND FUTURE WORK

Our research focuses on evolving personalized enjoyable levels for a maze runner game using an interactive genetic algorithm and cellular automata. The results show that we were able to evolve enjoyable maze runner levels. Limiting user fatigue by having them evaluate only a subsection of the population and only evaluating a small number of generations does not prevent the IGA from evolving good levels quickly.

This approach could be applied to different game genres like First Person Shooters, Real Time Strategy games, and Platformers to evolve levels with very different criteria for level fitness. A future approach that would allow for a high number of generations run and not increase user fatigue might be to have users evolve the fitness function rather than assigning the fitness directly.

REFERENCES

[1] Blizzard Entertainment. 1997. *Diablo*.
[2] Luigi Cardamone, Daniele Loiacono, and Pier Luca Lanzi. 2011. Interactive evolution for the procedural generation of tracks in a high-end racing game. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM, 395–402.
[3] Luigi Cardamone, Georgios N Yannakakis, Julian Togelius, and Pier Luca Lanzi. 2011. Evolving interesting maps for a first person shooter. In *European Conference on the Applications of Evolutionary Computation*. Springer, 63–72.
[4] Kate Compton and Michael Mateas. 2006. Procedural Level Design for Platform Games.. In *AIIDE*. 109–111.
[5] Vincent Hom and Joe Marks. 2007. Automatic design of balanced board games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*. 25–30.
[6] Martin Jennings-Teats, Gillian Smith, and Noah Wardrip-Fruin. 2010. Polymorph: dynamic difficulty adjustment through level generation. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*. ACM, 11.
[7] Lawrence Johnson, Georgios N Yannakakis, and Julian Togelius. 2010. Cellular automata for real-time generation of infinite cave levels. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*. ACM, 10.
[8] Sebastian Lague. 2015. Procedural Cave Generation Tutorial. (2015). https://unity3d.com/learn/tutorials/projects/procedural-cave-generation-tutorial
[9] Chris Pedersen, Julian Togelius, and Georgios N Yannakakis. 2009. Optimization of Platform Game Levels for Player Experience.. In *AIIDE*.
[10] Gillian Smith, Mike Treanor, Jim Whitehead, and Michael Mateas. 2009. Rhythm-based level generation for 2D platformers. In *Proceedings of the 4th International Conference on Foundations of Digital Games*. ACM, 175–182.
[11] Joshua R Smith. 1991. Designing Biomorphs with an Interactive Genetic Algorithm.. In *ICGA*. 535–538.