Difficulty Curve-Based Procedural Generation of Scrolling Shooter Enemy Formations

Pratama Wirya Atmaja¹, Sugiarto², Eka Prakarsa Mandyartha³

¹ ² ³ Department of Informatics Engineering, University of Pembangunan Nasional “Veteran” Jawa Timur, Jalan Raya Rungkut Madya, Surabaya, Indonesia

¹ pratama_wirya.fik@upnjatim.ac.id
² sugiarto.if@upnjatim.ac.id
³ eka_prakarsa.fik@upnjatim.ac.id

Abstract. In any game where the player must face large formations of enemies, the way the formations are formed significantly influences the difficulty level of the game. As proper difficulty levels are essential to player’s enjoyment, the enemy formations need to be arranged properly. In this paper we experimented with how to procedurally generate enemy formations in a vertical scrolling shooter game while keeping the difficulty levels proper. Generations of the formations were done with genetic algorithm where the fitness function calculated two variables: difficulty curve, which reflects a formation’s difficulty level, and enemy variety. The difficulty curve took into account difficulty points, which measured the danger levels of on-screen enemies throughout the game’s duration. The fitness of difficulty curve of an enemy formation was acquired through point-by-point comparison between the curve and a human-designed one, which represented the ideal difficulty curve as intended by the game’s developer, and calculation of the root-mean-square error. The genotype of an individual was represented by a 5x40 grid containing enemy genes and empty space genes. We ran 300 generations in total and each was done in 100 iterations and had 40 individual genotypes. The results show that the amount of enemy genes in initial genotypes influences the population’s fitness progression. We then discuss the implications of our findings and possible directions of future researches.

1. Introduction

Shoot-em-up is one of the oldest digital game genres, with the first game being Space Invaders (Taito Corp., 1978). The main appeal of the genre is its simple yet intense gameplay, which makes for a short yet spectacle-worthy entertainment. Despite its age, the genre is still thriving and new shoot-em-up games are still produced nowadays [1, 2]. The longevity of retro-style and 2D games is the main factor that helps the genre survive in 21st century.

One important concern in modern game development is cutting the development cost. This led to generating gameplay contents procedurally instead of by hand, saving the developer’s resources and even providing the players with much more vast contents to enjoy [3]. In order to produce contents with acceptable quality, the procedural generation has to be controlled, which heavily depends on the type of contents to be generated. To implement well-controlled procedural generation of game contents, therefore, the characteristics of the contents and the games have to be well-understood.

In this research we set to understand how to procedurally generate contents of a shoot-em-up game. The type of content we experimented on was enemy formations and we narrowed the shoot-em-up genre to scrolling shooters, a popular subgenre where the gameplay scene scrolls automatically.
1.1. Mechanics of Scrolling Shooters
A typical scrolling shooter game depicts the player’s character (most commonly a plane or spacecraft) traversing hostile environments full of adversaries [4]. Scrolling shooters is a subgenre of shoot-em-ups which is itself a subgenre of action games. Some common elements of action games are shared with scrolling shooters such as power-up items, lives system, smart bombs, and boss enemies [5]. On the other hand, scrolling shooters also exhibit unique characteristics, most importantly the one that gave the subgenre its name: the fixed scrolling system. Instead of influenced by the movements of the player’s character, the gameplay scene depicted on the screen independently scrolls vertically, horizontally, or both while various objects appear on the playfield from various angles [6]. The enemies typically come in formations (also called waves) exhibiting certain patterns. The formation of enemies in a scrolling shooter level usually determines the starting positions of the enemies. In a scrolling shooter game, the actual level that is being played is quite long or wide and the playfield only includes a portion of the level at one time. The enemies are generated idle in the level; as the playfield scrolls, it gradually overlaps with idle enemies, activating them. To preserve fun factor and gameplay balance, the player is prevented from shooting and damaging idle enemies outside of the playfield. An active enemy has a set amount of time for it to stay inside the playfield, which is usually determined indirectly by its movement pattern. As soon as the enemy moves out of the playfield, it immediately ceases to exist.

1.2. Procedural Content Generation
Development cost is a big concern for game developers, especially as digital game industry continues to grow enormously [7] and games continue to provide players with more and more contents. It is not surprising, then, that procedural content generation (PCG) has become one of the most active research topics in digital game field. Various PCG methods for various contents of various game genres have been found, such as levels of platformer games [8], levels of dungeon crawler games [9], levels of the popular puzzle game Angry Birds [10], and maps of strategy games [11]. Other than for gameplay purposes, PCG has also been implemented for generations of visual arts [12] and other non-gameplay contents.

Two previous researches of note on PCG for shoot-em-up games were done by Khalifa et al. [13] and Hastings et al. [14]. Hastings et al. experimented with using evolutionary algorithm to evolve player weapons in Galactic Arms Race, a shoot-em-up game where the player’s weapons get stronger and change their characteristics depending on how the player use them. Khalifa et al. experimented with a different subgenre of shoot-em-up, bullet hell, which tasks the player with surviving intense barrages of enemy bullets that may seem impossible to avoid at first glance. A constrained MAP-Elites algorithm was used to evolve a population of bullet barrage patterns in order to find the best and most fun pattern for the player.

1.3. Difficulty Curves
The notion of “difficulty curve” conveys the meaning of difficulty level progressing throughout the game or some parts of it. It is widely accepted that games need to have their difficulty levels progress in a certain way to match the player’s needs [15]. Various difficulty curves have existed in different commercial games: linear increase, linear increase with randomness, linear decrease, and so on [16]. Previous research by Qin, Rau, and Salvendy [17] concluded that the difficulty of a game level ideally should change up and down, which means the level starts out relatively easy, gradually gets harder toward the middle of the level, and gets easier again before rises for the second time to the end of the level. The ideal difficulty change is also mid-paced: the difficulty should rise and drop neither too fast nor too slow.

Traichioiu, Bakkes, and Roijers employed grammar-based approach to generate levels of a Legend of Zelda-like action-adventure game that satisfy difficulty curve constraints set by the game’s developer [18]. Adrian and Ana Luisa [19] generated the levels of a racing game with regards to difficulty curves within the levels. By reading the desired difficulty curves as numerical data and using the data in the fitness function, they were able to generalize their approach to cover every other game genre.
2. Research Methodology

In this research, we based our method on the work of Adrian and Ana Luisa [19] but we tailored their approach to fit the scrolling shooter genre. The playfield in our game scrolled vertically from bottom to top.

2.1. Spatial Enemy Formation Generation

A level in our game was 800x8000 pixels with two empty 800x1000 regions, one at the bottom and another at the top of the level. The remaining 800x6000 region contained a 5x40 grid where enemies would be generated in its 150x150 pixels cells. Each enemy occupied one grid cell and therefore a formation could contain at most 200 enemies. The 800x600 pixels playfield started at the bottom of the level, gradually moved upward two pixels per game frame.

2.2. Enemy Types and Difficulty Curve Calculation

We employed five enemy types in our experiment as seen in Figure 1. Every enemy had a “danger point” which measured how dangerous it was to the player. Characteristics of the five enemy types can be seen in Table 1. The danger point of every enemy was made constant for as long as the enemy was active.

The difficulty curve of an enemy formation was composed of a sequence of difficulty points, each the sum of danger points of all on-screen enemies at a specific game frame. Every difficulty curve would have 3700 difficulty points because the playfield required as many game frames to finish scrolling. To calculate the difficulty points, we moved the playfield from the bottom to the top of the level. Any enemy that came into contact with the playfield would be activated. The enemy’s danger point was taken into difficulty point calculation as long as it was active.

For simplicity in calculating a formation’s fitness, we later converted difficulty points to a range between 0 and 1. This means that the highest difficulty point possible had to be found as it would be converted to 1. Finding the maximum difficulty point was a matter of finding the arrangement of enemies that were possible to be active at the same time that would give the maximum point.

2.3. Generation Procedure

We generated enemy formations with genetic algorithm. The genotype of an individual was represented by a 5x40 grid of characters, which may consist of five types of enemy genes and one type of gene for empty space. We conducted 300 generation processes to generate 300 enemy formations, and every generation process was done in 100 iterations and its population contained 40 individuals.

![Figure 1](image)

**Figure 1.** Five enemy types in our experiment. From left to right, Enemy 1 to Enemy 5.

| Enemy type | Danger point | Active time (game frame) |
|------------|--------------|--------------------------|
| Enemy 1    | 100          | 100                      |
| Enemy 2    | 150          | 120                      |
| Enemy 3    | 200          | 90                       |
| Enemy 4    | 400          | 240                      |
| Enemy 5    | 650          | 150                      |

**Table 1.** Characteristics of enemy types
At the start of a generation process, the genotype grids of the population’s initial individuals would be filled with certain amounts of enemy genes. Any grid cells not occupied by enemy genes would automatically host empty space genes. In each iteration of the generation process, individuals in a population underwent either recombination, with 90% chance, or mutation. 16 individuals would become parents where half of them were of the highest fitnesses and the others were chosen randomly from the rest of the population. Recombinations were done via two-point crossover producing two offsprings. The resulting 16 offsprings may also mutate with 20% chance. When an individual mutated, one of three things may happen:

1) Its genotype gained 1 to 20 new enemy genes (replacing random empty space genes), not exceeding the number of the genotype’s grid cells;
2) Its genotype lost 1 to 20 random enemy genes (replaced with empty space genes);
3) 2 to 40 of its genes (selected randomly) switched their positions.

The fitness function calculated two fitness criteria: the difficulty curve and enemy variety in an individual. The difficulty curve fitness was acquired through point-by-point comparison between the individual’s difficulty curve and one we created manually. The manually-created curve, as seen in Figure 2, reflected the desired and ideal difficulty curve as proposed by Qin, Rau, and Salvendy [17]. Regarding enemy variety fitness, weaker enemies in scrolling shooter games generally appear more often than the stronger ones. We therefore set the ratio of Enemy 1 to Enemy 5 to 4:2:2:1:1. The more a formation violated this ratio, the lower its fitness would be.

2.4. Fitness Functions

Equation (1) was used to calculate the difficulty curve fitness of an enemy formation with \( n \) difficulty points. The difficulty curve fitness value \( f_r \) equals 1 minus the root-mean-square error (RMSE) between the desired difficulty points and the actual difficulty points of the enemy formation. \( ps_i \) is the desired difficulty point at game frame \( i \) and \( pa_i \) is the actual point, both range between 0 and 1.

The enemy variety fitness was calculated with (2). The fitness value \( f_v \) equals 1 minus the RMSE between the desired variety \( vs_i \) and actual variety \( va_i \) of \( t \) number enemy types. Because we used five enemy types, \( t \) was fixed to 5 and both \( vs_i \) and \( va_i \) also range between 0 and 1. To get the value of \( va_i \) of an enemy type \( i \), we calculated the sum of all enemies of that type in the formation and divided it by the total number of enemies in the formation.

The total fitness value of an individual was equal to the weighted sum of difficulty curve fitness \( f_r \) and enemy variety fitness \( f_v \). In this research we set the weight of \( f_r \) to 0.8 and the weight of \( f_v \) to 0.2 as we prioritized difficulty curve over enemy variety.

\[
fr = 1 - \sqrt{\frac{\sum_{i=1}^{n} (ps_i - pa_i)^2}{n}} \tag{1}
\]

\[
fv = 1 - \sqrt{\frac{\sum_{i=1}^{t} (vs_i - va_i)^2}{t}} \tag{2}
\]

Figure 2. The desired difficulty curve, based on the work of Qin, Rau, and Salvendy.
3. Results and Discussions

We run 300 generation processes in total which are split evenly into two groups: Group A uses the desired difficulty curve as seen in Figure 2, and Group B uses the same curve but with all its points divided by four. For each group we experiment with five ranges of initial enemy gene amount: 10 to 40, 30 to 70, 50 to 150, 130 to 170, and 160 to 190. We run 30 generation processes for each range, generating 150 enemy formations for each group, and we analyze the fitness progressions and the generated enemy formations.

3.1. Effect of Initial Enemy Gene Amount

Figure 3 shows fitness progressions of Group A throughout 100 generation iterations across five initial enemy gene amount ranges, with the y axis showing average total fitness values of 30 generation processes for each range. We can see that the amount of enemy genes of a population’s initial individuals influences the progression of the population’s fitness value. An initial individual with lower amount of enemy genes (within 10-40 or 30-70 range) will produce a sparsely-populated enemy formation grid with many empty rows, which yields a difficulty curve with many minimal and even zero difficulty points. Such difficulty curve of course differs strikingly from the desired one.

Figure 4 shows fitness progressions of Group B. Compared to those of Group A, we can see that the progressions are reversed between the five ranges of initial enemy gene amount: the more dense the initial enemy genes of an individual, the lower its fitness will be. The obvious reason for this is the reduced-to-one-fourth desired difficulty curve steers the generations toward much more sparse enemy formation grids, therefore favoring the 10-40 and 30-70 ranges. Another difference between the original desired difficulty curve and the reduced one is that the latter allows total fitness values to converge much sooner and the final values to be higher (0.97 on average, compared to 0.87 with the original desired difficulty curve). The probable reason for both the faster convergence and the higher final fitness values is that sparser enemy formations are easier to generate than denser ones. We are yet to know whether this is true only in the context of our experiment, or it can be generalized to every scrolling shooter game.

Figure 3. Effect of initial individual’s enemy gene amount on total fitness value progression in Group A.

Figure 4. Effect of initial individual’s enemy gene amount on total fitness value progression in Group B.
3.2. Generated Enemy Formations
For Group A, the final total fitness values range from 0.86 to 0.88. Only six out of 150 populations achieve the highest value (0.88); one such population is of 130-170 initial enemy gene amount range and its best initial individual and best final one can be seen in Figure 5 along with their difficulty curves.

It can be observed that Enemy 4 and 5, which possess the highest danger points, are generated in places where the difficulty curve rises. It seems that higher difficulty points in the desired difficulty curve would cause the amount of Enemy 4 and 5 in a population to rise, which would in turn violate the “weaker enemies appear more and stronger ones appear less” principle. This is evidenced by the higher total fitness values of Group B, which is partly because the group’s reduced-to-one-fourth desired difficulty curve allows the populations to achieve much better variety fitness.

![Difficulty curves and enemy formations](image)

Figure 5. Difficulty curves and enemy formations (rotated 90° clockwise) of the best initial individual (top) and the best final individual (bottom) of a Group A population with 0.88 total fitness value.

4. Conclusions
In this research we have experimented on using manually-defined data of desired difficulty curve to guide procedural generation of enemy formations in scrolling shooter games. The desired difficulty curve follows the ideal one proposed in literature and is used to get the fitness of an enemy formation by means of RMSE calculation between the curve and the formation’s difficulty curve, which is acquired through calculation of danger points of all on-screen enemies in the formation from the first to the last game frame. The fitness value also takes into account the formation’s enemy variety with regards to power differences between the enemies. Our results show that initial amount of enemy genes in a population’s genotypes influences the population’s fitness progression, and that maintaining the correct enemy variety strongly depends on the desired difficulty curve.

One possible direction for future researches is understanding other factors that may influence a population’s fitness progression, such as danger points and active times of enemies and various kinds of desired difficulty curves. Another direction is to explore different gameplay characteristics such as the inclusion of powerups, weapon and bomb pickups, and more complex enemy placement systems.
References

[1] S. M. Morin, "WHY SHMUPS ARE GREAT: WELCOMING BACK SHOOT EM’ UPS," 8 August 2018. [Online]. Available: https://www.keengamer.com/article/20177_why-shmups-are-great-welcoming-back-shoot-em-ups. [Accessed 6 August 2019].

[2] "What are the best Shoot 'Em Up games on Steam?," 10 February 2019. [Online]. Available: https://www.slant.co/topics/6383--shoot-em-up-games-on-steam. [Accessed 6 August 2019].

[3] N. Shaker, J. Togelius and M. J. Nelson, "Why use procedural content generation?," in Procedural Content Generation in Games, Springer International Publishing, 2016.

[4] "Shoot 'em up video games," 19 August 2018. [Online]. Available: https://gamicus.gamepedia.com/Shoot_%27em_up_video_games. [Accessed 6 August 2019].

[5] A. Rollings and E. Adams, "Action Games," in Andrew Rollings and Ernest Adams on Game Design, New Riders Publishing, 2003.

[6] L. McMillan, "All of Your Base are Belong to Us? Shmups as a Source for Better Game Design,” PhD Doctorate Thesis, Queensland Conservatorium of Music, Griffith University, 2011.

[7] J. Batchelor, "Games industry generated $108.4bn in revenues in 2017," Gamer Network, 31 January 2018. [Online]. Available: https://www.gamesindustry.biz/articles/2018-01-31-games-industry-generated-usd108-4bn-in-revenues-in-2017. [Accessed 6 August 2019].

[8] V. Volz, J. Schrum, J. Liu, S. M. Lucas, A. Smith and S. Risi, "Evolving mario levels in the latent space of a deep convolutional generative adversarial network,” in GECCO ’18 Proceedings of the Genetic and Evolutionary Computation Conference, Kyoto, Japan, 2018.

[9] A. Liapis, "Multi-segment Evolution of Dungeon Game Levels," in Proceedings of the Genetic and Evolutionary Computation Conference, Berlin, Germany, 2017.

[10] M. Stephenson and J. Renz, "Procedural generation of complex stable structures for angry birds levels," in 2016 IEEE Conference on Computational Intelligence and Games (CIG), 2016.

[11] J. Kowalski, R. Miernik, P. Pytlik, M. Pawlikowski, K. Piecuch and J. Sekowski, "Strategic Features and Terrain Generation for Balanced Heroes of Might and Magic III Maps," in 2018 IEEE Conference on Computational Intelligence and Games (CIG), 2018.

[12] A. Liapis, "Exploring the Visual Styles of Arcade Game Assets," in Proceedings of Evolutionary and Biologically Inspired Music, Sound, Art and Design (EvoMusArt), 2016.

[13] A. Khalifa, S. Lee, A. Nealen and J. Togelius, "Talakat: bullet hell generation through constrained map-elites," in Proceedings of the Genetic and Evolutionary Computation Conference, Kyoto, Japan, 2018.

[14] E. J. Hastings, R. K. Guha and K. O. Stanley, "Evolving content in the Galactic Arms Race video game," in 2009 IEEE Symposium on Computational Intelligence and Games, 2009.

[15] M. Lopez, "Gameplay Design Fundamentals: Gameplay Progression," 26 November 2006. [Online]. Available: https://www.gamasutra.com/view/feature/130188/gameplay_design_fundamentals_.php. [Accessed 6 August 2019].

[16] A. Nagle, P. Wolf and R. Riener, "Towards a system of customized video game mechanics based on player personality: Relating the Big Five personality traits with difficulty adaptation in a first-person shooter game," Entertainment Computing, vol. 13, pp. 10-24, 2016.

[17] H. Qin, P.-L. P. Rau and G. Salvendy, "Effects of different scenarios of game difficulty on player immersion," Interacting with Computers, vol. 22, no. 3, pp. 230-239, May 2010.

[18] M. Traichioiu, S. Bakkes and D. Roijers, "Grammar-based Procedural Content Generation from Designer-provided Difficulty Curves," in Proceedings of the 10th International Conference on the Foundations of Digital Games (FDG 2015), 2015.

[19] D.-F. H. Adrian and S.-G. C. Ana Luisa, "An approach to level design using procedural content generation and difficulty curves," in 2013 IEEE Conference on Computational Intelligence in Games, 2013.
