Genetic Algorithm-Based Movement Patterns for Scrolling-Shooter Games

Chang-Hoon Park, Jinseok Seo

***Abstract*— One of the exciting elements of scrolling-shooter games is finding the movement pattern of enemies, and predicting the direction in which they will move to eliminate them. However, if players identify all the movement patterns of the enemies, they will easily lose interest in game play. In the end, in today's game ecosystem, which has a very short life-cycle, players who feel weary will look for new games. To address this problem, this paper presents a genetic algorithm based technique for dynamically generating movement patterns of enemy aircrafts. The proposed method evolves the movement patterns designed by a game designer in the early stage to the more complex form of movement according to the player 's ability and vice versa.**

***Index Terms*— Game AI, Genetic Algorithm, PCG(Procedural Content Generation), Scrolling-Shooter Game.**

I. INTRODUCTION

Game designer Greg Costikyan said that the game is a decision making with enough information, and the player is going to go beyond the obstacles in front of him to achieve his goal [1]. That is, players play the game to reach the goal beyond the obstacle of the game of their choice. There, the game chosen by the player must provide the resources that the player can manage, the goal to achieve, and the obstacles that impede the achievement of the goal.

Players play the game repeatedly to achieve a given goal, and in the process, they repeat a pattern of obstacles or experiences. If the players are experienced and can gradually predict the near future, they will no longer be nervous [2]. When a player adapts to the game, he will no longer be interested in the game and will look for another interesting game. Continuing to create an unforeseeable situation for the player will reduce the same repeated experiences and prevent the game from being bored.

There are a few ways to create a situation that prevents players from predicting. First, the game developers should constantly design and update new contents. However, this method requires a lot of time and cost, and it cannot take all the characteristics of various players into account. The second way is to use random numbers for the creation of gameplay elements. This method is easy to implement because the generation algorithm is relatively simple. However, on the other hand, there is a disadvantage that game developers can not predict the contents. As a result, there is a limit to the amount and diversity of content that game developers can provide with the above method. One of the most actively studied areas to solve this problem is the Procedural Content Generation (PCG). PCG algorithms can generate game content dynamically by algorithms at runtime.

In this research, a sort of PCG technique is proposed to diversify the patterns of enemy aircrafts' movements in scrolling-shooter games. This paper presents a genetic based AI (Artificial Intelligence) algorithm that collects data about a player's gameplay results and enemies' movements, analyzes the data, and provides new movement patterns of enemies when the next level starts reflecting the analyzed results. The proposed method evolves the movement patterns designed by a game designer in the early stage to the more complex form of movement according to the player's ability and vice versa.

Chang-Hoon Park, Department of Digital Media, Dong-eui University, Busan, Korea.

Jinseok Seo, Division of Digital Contents Technology, Dong-eui University, Busan, Korea. (*corresponding author)

This work was supported by the ICT R&D program of MSIP/IITP (I5501-16-1016, Instant 3D based Join & Joy content technology).

II. RELATED WORKS

It is quite natural that it takes a lot of time and effort to produce rich and high-quality game contents that meet the needs of various users. In addition, the larger the game market, the more demand for content will increase and the greater the scalability of content. Recently, procedural content generation (PCG) techniques using various kinds of algorithms have been studied to solve this problem.

The primary purpose of PCG is automating or helping to create various content within a game. However, there are many practical difficulties to apply PCG to a game. An example of a typical difficulty is that the content generated using PCG should be artistically complete, satisfy the requirements of the artist, and be interesting to users at the same time.

At Delft University in the Netherlands, for the first time, the PCG field was investigated and summarized in depth [3]. They divided the game content into 6 layers depending on the depth of the object to which an algorithm applies. These layers have low-level contents such as textures and sounds as you go down the content layer pyramid, and more abstract contents such as maps, environments, stories, and leaderboards is placed on top. Based on these layers, they investigate layers the existing PCG techniques are applied to, and provide guidelines for using the PCG technique. In addition, this research concludes that the PCG technique applied to one layer can be applied to other layers as well. In this research, we tried to apply the genetic algorithm used in various fields to the movement pattern of enemy aircrafts in a scrolling-shooter game.

Shaker et al. [4][5] conducted researches to automatically generate various types of platform for the optimal experience of game players in the "Infinite Mario Bros" game released as open source. They defined the elements representing the game level and the features of a player's gameplay as variables and predicted the user 's game experience using sequence mining and neural network. An evolutionary

algorithm was used to find level designs that maximize a desired effect among challenge, engagements, and frustration.

ITU's Liapis et al.[6] automatically created the mesh models for space crafts by applying the FI-2Pop (Feasible-Infeasible Two-Population) genetic algorithm to the CPPN-NEAT (Compositional Pattern Producing Network - NeuroEvolution of Augmenting Topologies) algorithm. In order to define the function for evaluating the generated mesh, symmetry, weight in the bottom half, weight in the middle third along the x-axis, weight in the middle third along the y-axis, containment within a forward-point triangles, simplicity, and jaggedness are quantified.

Togelius et al. proposed a method for automatically evolving tracks in racing games [7] and a method for automatically generating maps in a real-time strategy game, Starcraft [8]. In addition to above researches, we can find many PCG algorithms applied to various games [9][10][11][12]. As can be seen from most of the researches, the commonly used methods for automatically generating content in games are based on various artificial intelligence techniques. In the future, as the game industry develops and virtual reality contents become popular, more PCG algorithms based on artificial intelligence techniques are expected to be applied.

III. PCG for Scrolling-Shooter Games

A. *Test Platform: "Xhootings" Game*

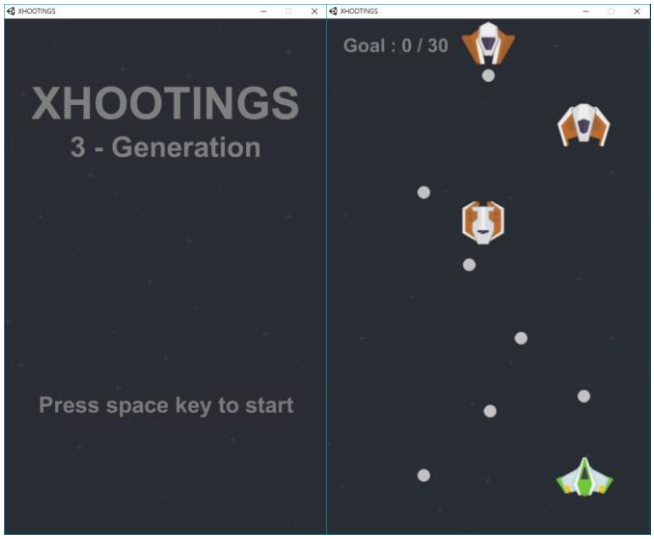

Figure 1: Screenshots of "Xhootings" game

In order to verify the proposed PCG algorithm, we developed a simple scrolling-shooter game called "Xhootings" (Fig. 1). When the game starts, the enemy airplane, which is infinitely generated at intervals of 1.3 seconds, moves down, and the objective of this game is to attack these enemies by shooting bullets. The game collects various data needed for evolution as the user plays the game. When the game is started, the user deals with enemy airplanes which are moving with patterns created by a game designer. The first level starts with 6 types of basic movement patterns. At each level, users play until they kill 30 enemies, which are destroyed when they are hit three times by bullets. The proposed PCG algorithm collects four types of data during game play, which includes the user's play time, the number of player's airplanes used by the user during the play, the number of times each enemy has destroyed the player's aircraft, the number of times each enemy has been destroyed by the user.

B. *Evolution Algorithm*

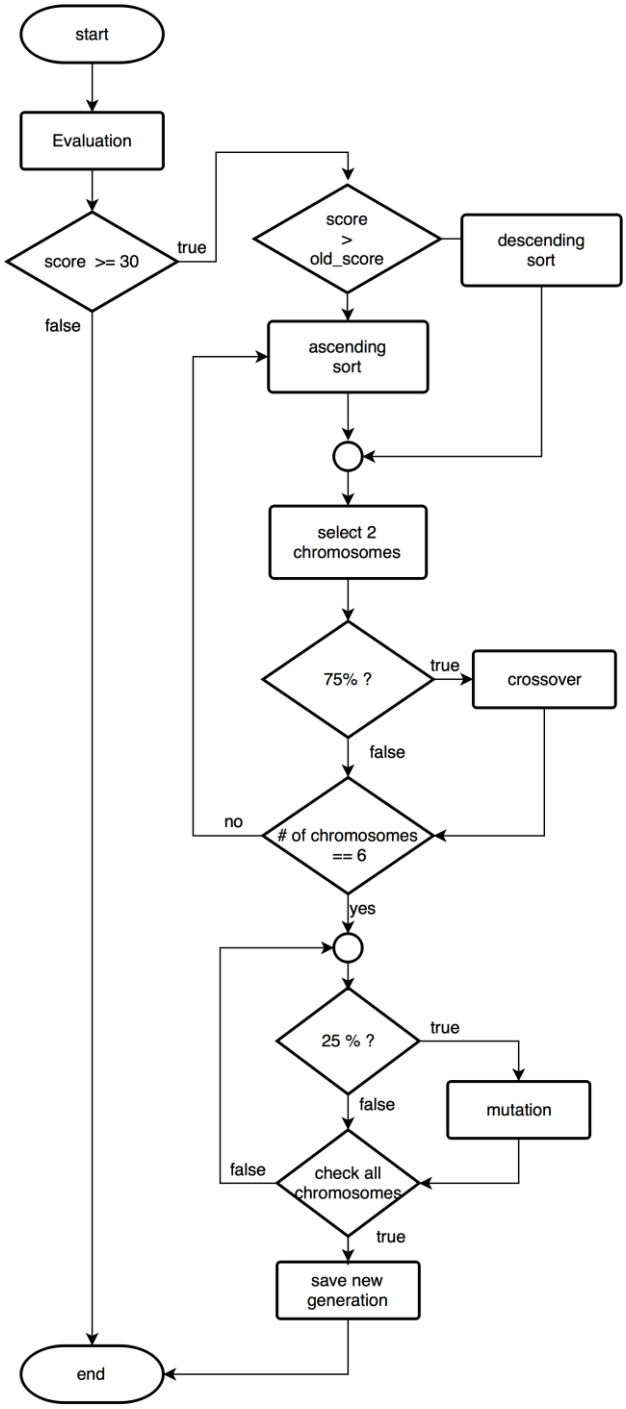

Figure 2: Overall process of our evolution algorithm

When the user reaches a given goal (to kill 30 enemies) and ends one level, the PCG system performs a chromosome evolutionary algorithm to create new movement patterns of enemies by using the collected data. The overall process of our evolution algorithm is depicted in Fig. 2. The chromosome evolution process proceeds in three stages: evaluation, crossover, and mutation.

In the evaluation stage, the user's score and the fitness function values of all types of enemies are calculated based on the data collected during the previous game play. The player's score determines the progress of the evolutionary algorithm

and the order in which the chromosomes are sorted, and the enemies' fitness function values are used to choose a pair of chromosomes for the crossover stage. The crossover stage proceeds until all the chromosomes of one generation are generated. Since the number of chromosomes used in this study is 6, crossover occurs up to 3 times. When the crossover stage is finished, the next step, mutation stage, proceeds. When the mutation process is complete, the newly created 6 kinds of enemies appear at the next level.

1) Encoding Chromosome

The original genetic algorithm encodes the gene chromosomal information into a bit stream, but in this study, it is encoded as an array of real numbers because the data size required for the operation is not only small but also the calculation process is very simple.

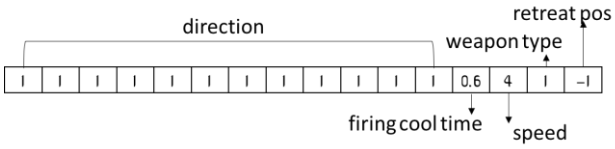

Figure 3: An Encoded Chromosome

The chromosome of the enemy represents the total of 5 kinds of information by genes, including moving direction, delay time of bullet launch, speed of movement, weapon type, and retreat position of an enemy. In the case of the direction of movement, for example, each gene is represented by -1, 0, or 1, which indicates the direction in which the enemy moves while falling down (Table 1). In the case of the chromosome in Fig. 3, the enemy continues to move to the right and falls down.

Table 1: Values for Gens for Movement Direction

Value	Meaning
1	move right
0	in place
-1	move left

2) Evaluation

As mentioned above, in the evaluation stage, the fitness function value of each enemy is calculated together with the score of the user's play. First, the score for the user's play is calculated by taking the number of airplanes used by the user and the play time as parameters. Naturally, we defined the score function so that the more plyer's aircrafts are used and the longer the play time, the lower the score (Equation 1).

$$player\_score = \frac{1.2}{num\_of\_player\_aircrafts \times play\_time} \qquad (1)$$

In this research, if the player score is less than 30, the evolutionary algorithm is not performed. If the score is greater than 30, the evolution direction is changed differently by comparing the score of the current level with that of the previous level. If the player's score improves, the enemy evolves more strongly and vice versa.

The fitness value is calculated based on the number of times each enemy has destroyed the player's aircraft and the number of times the enemy has been shot down by the player during play (Equation 2).

$$fitness = \frac{player\_death + 1}{enemy\_death + 1} \qquad (2)$$

The chromosomes whose fitness values have been calculated are sorted in different order according to the evolution direction determined by the player's play score. The reason is that in order to evolve the enemy to be stronger, the chromosomes are sorted in descending order, and in the opposite case, they are sorted in ascending order so as to differentiate the probability of selecting a pair of chromosomes in the next step of crossover.

3) Crossover

In this research, one generation consists of 6 chromosomes. Therefore, in this stage, 6 new chromosomes for a new generation, whether the crossover operation is performed or not, are generated. We set the probability of applying the crossover operation to 75%.

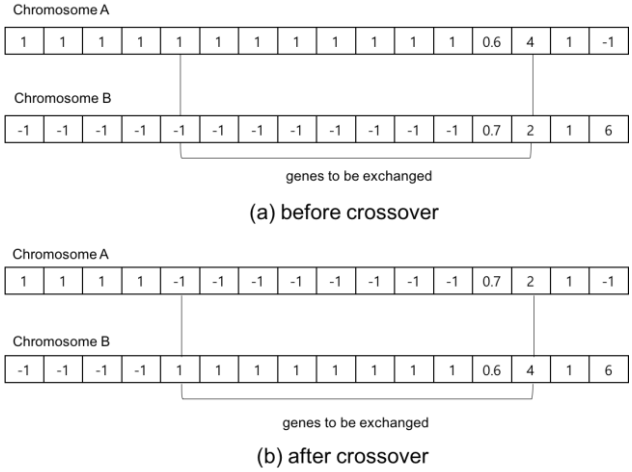

Figure 4: An Example of the Crossover Operation

Traditional genetic algorithms arbitrarily select the point at which the chromosome breaks (crossing point), and exchange all the genes behind this point on the chromosome. However, in this research, we proposed an expedient encoding scheme, so our PCG algorithm chooses randomly both starting and ending positions and exchanges the genes between them.

4) Mutation

In the conventional genetic algorithm to find the optimal solution, the mutation rate of each gene is usually set to 0.1%. However, in this research, an anomalous form of mutation algorithm was used in consideration of the characteristics our proposed encoding method.

First, the mutation probability of each chromosome was set to be 25%. Next, as in the crossover operation above, the starting and ending positions of the chromosome to which the mutation operation would be applied were arbitrarily determined. In addition, since the meanings of each gene are very different from each other, the mutation method for each gene is also designed differently.

Table 2: Mutation Range of Variable Types of Genes

Gene Type	Variation Range	Limit Range
direction	-1, 0, 1	-1, 0, 1
firing cool time	-0.3 ~ 0.3	0.6 ~ 1.0
speed	-3 ~ 3	3 ~ 7
weapon type	-1, 0, 1	0, 1, 2
retreat pos	-4 ~ 4	-1or 5 ~ 11

Table 2 summarizes the variation range and limit range of each gene. For example, the range of meaningful values of the gene for “direction” of movement is 0.6 to 1.0, and the range of values that can be varied in the variation calculation is any value between -0.3 and 0.3. If the mutation results out of the "limit range", a compensation operation is also performed so that it remains within a meaningful "limit range.” This process is shown in Fig. 5 as an algorithm, and the compensation operation is designed differently for each gene so that it is mutated to a meaningful value.

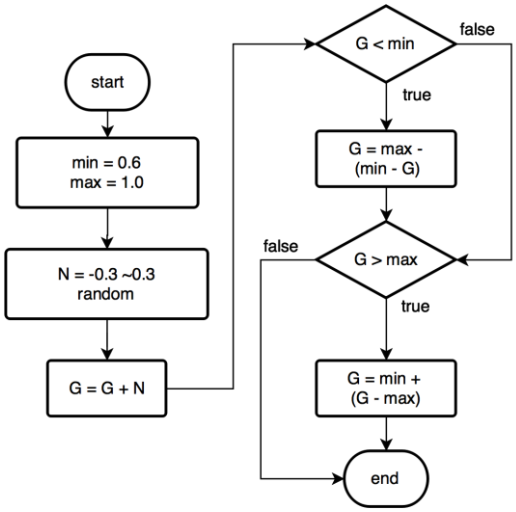

Figure 5: Algorithm for the Mutation and Compensation

5) *New Generation*

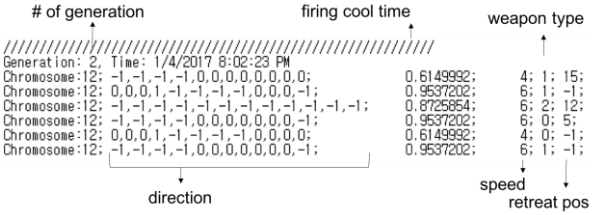

Figure 6: Raw Data of 1st Generation

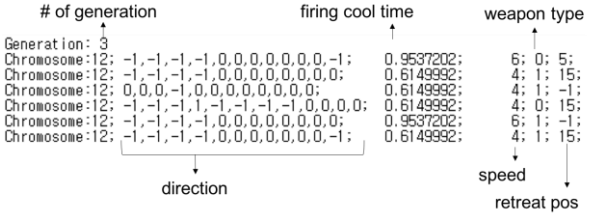

Figure 7: Raw Data of 2nd Generation

As mentioned above, in this research, we used a simple modified form of the genetic algorithm without using the traditional genetic algorithm designed to find the optimal solution. As a result, even if only one cycle of evolution algorithm is applied, it is possible to visually discover the creation of an enemy that shows a new shape e of movement. Fig. 6 and Fig. 7 show raw data representing the chromosomes of the second and third generations, respectively.

IV. Results

In order to verify how our proposed PCG algorithm generates effective movement patterns, we conducted experiments with two users. One is very experienced in scrolling-shooter games and the other is very inexperienced. It is very easy to find the enemy that changes its movement pattern even after the end of one level, but the evolved results to the tenth generation are summarized in Fig. 8 to show more dramatic results.

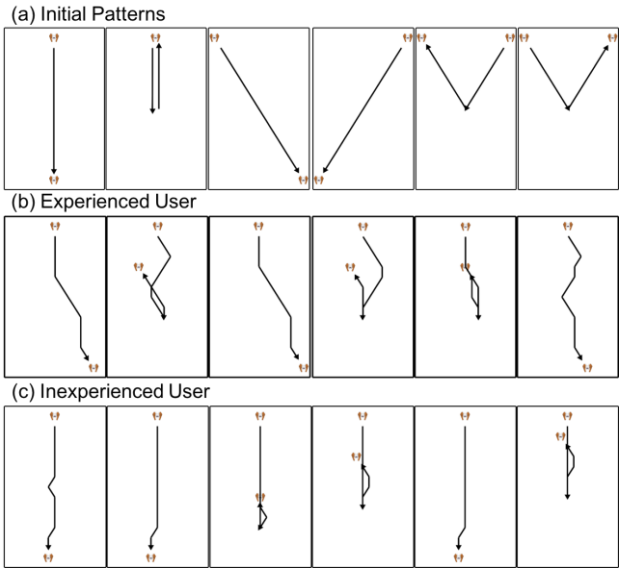

Figure 8: (a) Initial patterns, (b) patterns evolved into complex forms, (c) patterns evolved into simple forms

Experienced users acquire the ability to quickly adapt and evade even the enemy moving in a complex pattern. Thus, as the evolution proceeds over and over again, more complex patterns appear and the game becomes more difficult. On the other hand, inexperienced users take a long time to adapt to a slightly modified pattern. In this case, our algorithm has a higher probability of selecting chromosomes with lower fitness values, so that even if the same number of evolution is repeated starting from the same initial patterns, more simple patterns appear compared to experienced users.

V. Conclusion

In this paper, we proposed a novel movement pattern generation method for scrolling-shooter games using genetic algorithm. The proposed algorithm generates new movement patterns according to the user 's play results. In other words, it was implemented to give more complex missions to users who are clearing too easily, and to give easier missions to inexperienced users.

As can be seen from the above results, applying PCG algorithms to the game makes it possible to generate contents with the difficulty level suitable for users in real time. In addition, game designers can easily create and test various contents in advance. Although PCG algorithms is not yet capable of replacing all the contents that humans design, it is expected that various PCG algorithms will be able to replace more parts in the near future.

Acknowledgment

This work was supported by the ICT R&D program of MSIP/IITP (I5501-16-1016, Instant 3D based Join & Joy content technology).

REFERENCES

[1] G. Costikyan, "I Have No Words & I Must Design." Interactive Fantasy #2, 1994.
[2] C. Jeong, J. Ham, and J. Park, "Applying of SOM for Recognition to Tension and Relaxation in a Scrolling-Shooter Game," Proceedings of the Korean Society of Computer Information Conference 16(2), pp.169-172, 2009.
[3] H. Hendriks, S. Meijer, J. V. D. Velden, and A. Iosup, "Procedural Content Generation for Games: A Survey," ACM Transactions on Multimedia Computing, Communications and Applications, Feb., 2011.
[4] N. Shaker, G. N. Yannakakis and J. Togelius. Crowd- Sourcing the Aesthetics of Platform Games. IEEE Tran- sactions on Computational Intelligence and AI in Games, issue. 99, December, 2012.
[5] N. Shaker, G. Y. and J. Togelius. Digging deeper into platform game level design: session size and sequential features, in Proceedings of EvoGames: Applications of Evolutionary Computation, Lecture Notes on Computer Science, 2012.
[6] A. Liapis, G. N. Yannakakis, and J. Togelius, Adapting Models of Visual Aesthetics for Personalized Content Creation, IEEE Transactions on Computational Intelli- gence and AI in Games, Special Issue on Computatio- nal Aesthetics in Games Vol. 4, No. 3, pp. 213-228, Sep. 2012.
[7] J. Togelius, R. D. Nardi, and S. M. Lucas. Towards Au- tomatic Personalized Content Creation for Racing Games. IEEE Symposium on Computational Intelli- gence and Games. 2007.
[8] J. Togelius, M. Preuss, N. Beume, S. Wessing, J. Ha- gelback, and G.N. Yannakakis. Multiobjective Explo- ration of the StarCraft Map Space. In: Proceedings of the IEEE Conference on Computational Intelligence and Games. 2010.
[9] C. Grappiolo, Y.-G. Cheong, J. Togelius, R. Khaled, G. N. Yannakakis. Towards Player Adaptivity in a Serious Game for Conflict Resolution. In the Proceedings of VS-Games 2011 Natural Interaction and Player Satis- faction in Games Workshop. Athens, Greece. May 4-6, 2011.
[10] J. Hastings, R. K. Guha, and K. O. Stanley. Automatic content generation in the galactic arms race video game. IEEE Transactions on Computational Intelligence and AI in Games, 1(4):245-263, 2010.
[11] S. Risi, J. Lehman, D. B. D'Ambrosio, R. Hall, and K. O. Stanley. Combining Search-based Procedural Content Generation and Social Gaming in the Petalz Video Game. In: Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference(AIIDE), Menlo Park, CA. 2012.
[12] E. McDuffee and A. Pantaleev. Team Blockhead Wars: Generating FPS Weapons in a Multiplayer Environment. FDG PCG workshop 2013.

Chang-Hoon Park received the B.S. degree in Game Engineering from Dong-eui University, Korea, in 2016, and is currently in the M.S. course in Digital Media in Dong-eui University. His current research interest is artificial intelligence techniques for computer games.

Jinseok Seo received the M.S. and Ph.D. degrees in Computer Science and Engineering from Postech, Korea, in 2000 and 2005, respectively. Since 2005, he joined the division of digital contents technology, Dong-eui University, Busan, Korea. His main research interests are artificial intelligence for computer games, game engines, virtual reality, and augmented reality.