2003 International Symposium on Advanced Intelligent Systems
September 25-28, 2003, Jeju, Korea

Generation of Emergent Game Character's Behavior with Evolution Engine

Jin-Hyuk Hong and Sung-Bae Cho

Dept. of Computer Science, Yonsei University
134 Sinchon-dong, Sudaemoon-ku
Seoul 120-749, Korea
{hjinh, sbcho}@cs.yonsei.ac.kr

Abstract - In recent years, various digital characters, which are automatic and intelligent, are attempted with the introduction of artificial intelligence or artificial life. Since the style of a character's behavior is usually designed by a developer, the style is very static and simple. So such a simple pattern of the character cannot satisfy various users and easily makes them feel tedious. A game should maintain various and complex styles of a character's behavior, but it is very difficult for a developer to design various and complex behaviors of it. In this paper, we adopt the genetic algorithm to produce various and excellent behavior-styles of a character especially focusing on Robocode which is one of promising simulators for artificial intelligence.

I. INTRODUCTION

With the development of video games, the interest on intelligence for the game character is rapidly increased in recent years [1]. The techniques for the game move to the development of good intelligence, while they just focused on the process of graphics in its early stage. Many players prefer much higher level of intelligence, and the artificial intelligence of a game gets to be one of crucial factors for the success of the game [2].

Artificial intelligence controls the behavior of a character or the environments to be more realistic and attractive [3,4]. The research on the generation of character's behavior starts on robotics which controls the behavior of a robot. Recently with the growth in computer animations and various game industries, it also includes the studies to control the behavior of an avatar or an animation character [1]. While robotics focuses on the automation and accuracy of a robot's behavior, the research of character's behavior for the entertainment considers the diversity, creativity and aesthetics of the behavior as important.

Hence a digital character should not repeat the same behavior, but have various and creative behaviors to provide player with news and curiosity. In particular, game character should act diverse behaviors for the various environments and be designed not to be easily understood by a player. However traditional character is designed by human, so the behavior is not complex for a player to be understood. In addition, when the game's environment is complex, the behavior becomes much more complex to be designed. The traditional approaches are hard to develop diverse and remarkable behaviors to fill up the player's requests [1,2,5].

In this paper, we focus on the generation of diverse and good behaviors of a character. The genetic algorithm is used for the purpose to create emergent behaviors by combining primitive behaviors of a character.

II. RELATED WORKS

A. Behavior strategies of characters

Game player controls a virtual character to move and act in the virtual environment. Since the time for calculating a behavior should be short and the process should be relatively easy at a game, the finite state machine is commonly adopted to design the behavior of a game character. The finite state machine is usually constructed with 10 or fewer states. At each state the behavior of a character is defined statically with simple operations to process fast. As mentioned before, however, the style or strategy of a character's behavior is easily understood because of its simplicity [2].

There are many other artificial intelligent techniques for design of a character's behavior, such as case-based reasoning, decision tree, neural network, fuzzy logic, and so on [1,4]. Since most of them are based on manual design, they leave developer some difficulties to design the behavior in every case. The manual design is apt to restrict the diversity and creativity of a character. For the case of neural network, it does not need to design the behavior directly, but the inputs and outputs are explicitly defined so as to hardly expect extra behaviors of a character. In a

game, extra behaviors of a character cause an interest to the player, however it is hard to construct various and extra styles of the behavior with these traditional approaches. In order to solve these limitations, the genetic algorithm is applied to some games but not to various kinds of the computer games [5].

Artificial life is one of promising recent techniques for the design of a character's behavior [3,4]. It is a study of systems which imitate the behavioral characteristics of a living thing. It considers an 'emergent behavior', which is the result of interactions among low-level of actions, as important. Artificial life technique divides a huge AI which decides a character's behavior into a set of small actions so as to generate a complex and whole behavior with interactions between low-level of simple rules. Usually it is called as 'emergence'. Artificial life is a useful technique to generate complex and diverse behaviors of a character, even if it is based on primitive actions. In this paper, we define primitive actions of a character and combine them using the genetic algorithm to generate various styles of a character's behavior.

B. Robocode [7]

In order to demonstrate the proposed method, we adopt Robocode which arouses an interest recently as battle simulator for artificial intelligence. It is Java-based tank battle simulator developed by IBM Alphaworks. A tank programmed by user makes a move in the battle field and evades the opponent's attack and assualts it. As visualized simulator, it is easy to observe the behavior of a character. In addition, since it is currently on-line, it can be possible to compare various other tanks that are programmed by the others.

A Tank in battlefield should be survived from a battle among the opponents. It gathers various information on itself and the opponents, and decides its behavior with the information. It is necessary to consider complex and various factors to win a game and to execute the proper strategy of a character's behavior. Movement against the opponents and the efficient usage of an energy should be also considered, and the success of an attack leads to the win. Since a user manually designs and programs a tank, it acts accurately but is apt to have a simple behavior.

A tank starts a game with a limited energy, and it consumes the energy by shooting bullets or bumping somethings. If the energy becomes 0, the tank gets to be destroyed. The energy can be increased as a compensation when it hits the other tanks.

A tank consists of gun, radar, and body as shown in Fig. 1. Each part can operate independently. The gun fires a bullet, and the radar scans opponents and obtains their information, and the body is charge of movement. Each part has various operating functions.

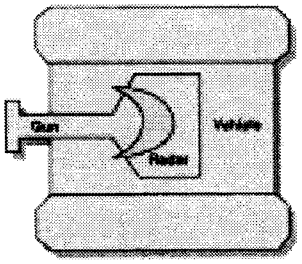

Figure 1. Structure of Robocode's tank

A tank has a basic behavior for its main loop, and additional behaviors for events which happen in some predefined situations. Common events used for designing a tank are ScannedRobotEvent (when radar scans a robot), HitByBulletEvent (when the tank is hit by a bullet), HitRobotEvent (when the tank collides with a robot), HitWallEvent (when the tank bumps into a wall), and so on. First, a tank acts its basic behavior, and when a specific event occures, the tank changes its behavior corresponding to the event.

III. EVOLVED STRATEGIES GENERATION

A. The genetic algorithm

The genetic algorithm is one of evolutionary computations based on the evolution theory such as genetic programming, evolutionary programming, and evolution strategies, which applies in optimization and classification problems [4,6]. There are some differences in the representation of an individual and in the evolutionary operations, but the fundamental principle and methods of them are the same. In this paper, we generate the strategies of a game character's behavior using the genetic algorithm.

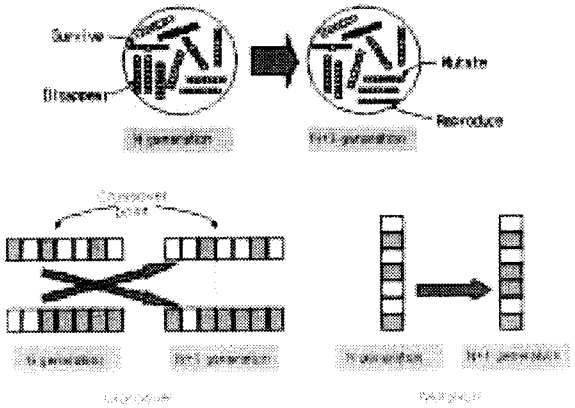

Figure 2 Genetic algorithm

The genetic algorithm, suggested by John Holland in the beginning of 1970s, is an optimization method imitating the mechanism of nature's evolution such as crossover, mutation, and the survival of the fittest. Since it provides an efficient search method based on a population, it applies in many problems for optimization and classification. Common procedure of genetic algorithm is as follows, while it uses genetic operators as shown in Fig. 2.

- step 1: initialize the population
- step 2: evaluate each individual's fitness of the population
- step 3: generate new population in proportion to the fitness of each individual
- step 4: execute genetic operators such as crossover and mutation
- step 5: repeat steps 2~5 until the stop condition is satisfied

The genetic algorithm is very efficient for general problems and provides an optimization mechanism, so that it applies to the optimization of gas pipe-line arrangement, travel salesman problem, the behavior evolution of robots, learning neural network, and the optimization of a fuzzy membership functions, and so on.

B. Encoding characters

The behavior of a tank is classified into move-strategy (MS), shoot-strategy (SS), bullet-power-strategy (BPS), radar-search-strategy (RSS), and target-select-strategy (TSS), all of which are predefined in this paper. Each strategy has various kinds of basic actions as shown in Table 1, and Table 2 sets some examples for move-strategy. The design of these basic actions is mostly simple for a developer to understand and implement them. Even if they are very simple for each, the composite usage of them generates more various and complicated behaviors.

Table 1 Basic actions of a tank

MS	SS	BPS	RSS	TSS
random	constant	distance-based	always-turn	weak-robot
simple linear	linear	light-fast	target-focus	focus-attack
random linear	complex-linear	powerful -slow	target-scope-focus	defense
simple circular		medium		nearest-robot
random circular		hit-rate-based		
anti-gravity				
stop				
Bullet-avoid				

A tank basically moves on a behavior defined in main loop. When an event happens, the specific behavior for the event is generated. In this paper, we design a behavior for main loop and behaviors for each event. Especially, we consider typical 4 events mentioned previously. Each behavior has 5 kinds of strategies as shown in Table 1 to be decided. The style of these behaviors is generated by the genetic algorithm which uses the basic actions and searches the optimal combination of them. There are 7200 (=5*8*3*5*3*4) possible combinations of the basic actions. It is not quite large to search, but the complexity increases rapidly when the number of the basic actions increases.

Table 2 Example-codes of move-strategy

Move-strategy	Code

Random	switch(Math.random()*2){
	case 0: setAhead(Math.random()*500);
	break;
	case 1: setBack(Math.random()*500);
	break; }
	switch(Math.random()*2){
	case 0: setTurnRight(Math.random()*90);
	break;
	case 1: setTurnLeft(Math.random()*90);
	break; }
	execute();

Linear	ahead(100); setBack(100);

Circular	setTurnRight(5000);
	setMaxVelocity(5);
	ahead(5000);

IV. EXPERIMENTS

A. Experimental environment

We verify the proposed system with several tanks provided by Robocode such as wall-Robot, corner-Robot, and fire-Robot. Wall-robot maintains itself following a wall, and corner-Robot stays at a corner of a battle field, and fire-Robot stays a place and fires a bullet when it scanned an opponent. In addition, we evolve a robot against BigBear programmed by Daniel Johnson, which is one of the champs of Robocode Rumble in 2002 [7].

The parameters of the experiment are set as shown in Table 3. For genetic operation, 1 point crossover, bit-flip mutation, and elitism are used. Roulette wheel selection is adopted as selection mechanism. The fitness of each individual is measured by the result of 3 battles among the target tanks.

Table 3 Parameters of experimental environment

Genetic operator	Value
Population size	25
Maximum generation	100
Selection rate	0.6
Crossover rate	0.6
Mutation rate	0.1

B. Results of evolved strategies

In this paper, we focus on the generation of various and emergent behaviors using the genetic algorithm. The behavior should be also good enough to compete with the target tanks.

Table 4 shows the result of the experiment. Evolving a tank against corner-Robot is not difficult as you see, because corner-Robot has a simple behavior pattern. Fire-Robot and wall-Robot have a more complex pattern, so the system searches fewer solutions within the same generations. Because the system did not find any solution against BigBear in 50 generations, we have tried 150 generations to search solutions and it generates a few of good strategies to defeat BigBear. The values of the right two rows of the table are different, which indicates that good strategies are different corresponding to target tanks.

Table 4 Results of the system against target tanks

Target tank	Number of discovered behaviors (150 generations)	Most frequent behavior	
		Main loop	Scanned-RobotEvent
Corner	23	000310	001003
Fire	13	000200	012413
Wall	7	400302	412223
BigBear	0 (4)	000313	011112

Table 5 Emergent strategies against wall-Robot

Emergent strategy	Content	Value of affecting behavior
Rambo	Firing at random	012312
Master shot	Tracing the opponent accurately and shooting	412422
Bumping king	Not shooting, only bumping the opponent	401022
Runaway	Avoiding the opponent's attack until the opponent becomes exhausted	210003
Upset driver	Moving the battle filed incomprehensibly	012112

The proposed system has generated many interesting strategies. There are many different ways to defeat wall-Robot, and Table 5 shows some of the outstanding emergent strategies beating wall-Robot. Specially bumping king and runaway strategies are not under our expectation. The last row indicates the critical values for the behaviors.

V. CONCLUSIONS

Artificial life and the genetic algorithm are useful to generate various styles of a character's behavior. A game character should keep many strategies of its movement, but it is a difficult problem for a developer to design diverse behaviors. In this paper, we have tried to generate many interesting strategies of a game character using the genetic algorithm. Especially applying to Robocode, we have demonstrated its utility to design various behavior styles.

We used a set of predefined actions and combined them to generate the high level of behavior. As a future work, we will use the genetic programming, which is more flexible and representative than the genetic algorithm, to generate the structure of behavior and for better movement to calculate the proper value of many parameters such as distance, degree, power, etc.

ACKNOWLEDGEMENTS

This paper was supported in part from Korea Culture & Contents Agency.

REFERENCES

[1] J. Laird and M. Lent, "Human-level AI's killer application: Interactive computer games," *AI Magazine*, vol. 22, no. 2, pp.15-26, 2001.

[2] S. woodcock, "Game AI: the state of the industry," *Game Developer Magazine*, pp. 28-35, August 1999.

[3] C. Langton, C. Taylor, J. Farmer, and S. Rasmussen, "Artificial life 2," *in Sante Fe Institute Studies in the Sciences of Complexity, Addison-Wesley*, pp. 511-547, 1992.

[4] M. Mitchell and S. Forrest, "Genetic algorithms and artificial life," *Artificial Life*, vol. 1, no. 3, pp. 267-289, 1994.

[5] D. Johnson and J. Wiles, "Computer games with intelligence," *IEEE Int. Fuzzy Systems Conf.*, pp. 1355-1358, 2001.

[6] K. Chellapilla and D. Fogel, "Evolution, neural networks, games, and intelligence," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1471-1496, 1999.

[7] S. Li, Rock 'em, sock 'em robocoe!, 2002. http://www-106.ibm.com/developerworks/java/library/j-robocode