Game Player Modeling Using D-FSMs

Tae Bok Yoon, Dong Moon Kim, Kyo Hyeon Park, Jee Hyong Lee*, and Kwan-Ho You

School of Information & Communication Engineering
Sungkyunkwan University, Suwon, Korea
{tbyoon, skyscrape, megagame}@skku.edu,
{jhlee, khyou}@ece.skku.ac.kr

Abstract. Recently, various ways are being explored for enhancing the fun of computer games and lengthening the life cycle of them. Some games, add realistic graphic effect and excellent acoustic effect, and make the tendencies of game players reflected. This paper suggests the method to collect and analyze the action patterns of game players. The game players' patterns are modeled using FSM (Finite State Machine). The result obtained by analyzing the data on game players is used for creating NPCs (Non-Player Characters) which show new action patterns by altering the FSM defined previously. This characters are adaptable NPCs which is learnable the action patterns of game players. The proposal method can be applied to create characters which play the role of partners with game players or the role of enemies against game players.

1 Introduction

User Modeling is the user cognitive process in which the data on users occurred under a specific system environment are collected and analyzed to achieve the objective of the system more effectively [1]. User modeling technique is being exploited diversely to analyze the profile information of users in the many fields such as games, educations, ubiquitous technologies and web mining [2].

Particularly in the game environment, the services based on analyzing gaming data of players, collected in the process of game activity, and utilizing the user modeling technique is being researched actively. In the game 'Black & White by Peter Molyneux', a partner NPC named Creature, has a different appearance in accordance with the game operation of the game player, and in the game Sims, an NPC also has a different appearance according to the player's game operation[3]. However, until now, in the field of games, user modeling techniques which have been used in console games and package games, rely on specific features of individual games.

This paper suggests D-FSM (Dynamic Finite State Machine) method which can create new NPCs behavior patterns to which the game players` tendencies are reflected. This method alters the FSMs for NPCs which were set with static action patterns in the initial phase of games, by adding information on the initial FSMs based on the data on game players which are collected while of gaming. In this paper, for

* Corresponding author.

analysis of the collected data, decision tree method which is a kind of machine learning methods is used; the initial FSMs are altered with the analysis result. Along with this, script languages which set action patterns are defined so that D-FSM can be used flexibly for games in various categories D-FSM suggested in this paper can be used for following cases:

First, the suggested D-FSM can be used as a tool for designing NPCs in the initial phase of game production. Ordinarily game developers spend a lot of time and efforts to create NPCs in the initial phase of game production, in creating NPCs` behavior patterns. In designing computer games, if real game players` information is used, it will be much easier to create NPCs and thus the developing time and effort will be reduced. Moreover, it is possible to create more realistic NPCs of which behaviors are similar to human players.

Second, the suggested D-FSM can be used for creating partner NPCs in games. There are cases where NPCs share the same objectives with game players and cooperate with them. Since the D-FSM can build NPCs from human players' play record, NPCs which has abilities similar to game players and play games can easily be created.

Third, the suggested D-FSM can be used for creating hostile NPCs in games. Since the action patterns of monster NPCs in most games are also static, human players easily penetrate the patterns. This fact can be a factor which dwindles the life cycle of the games. The model dynamically generated from human players' play record can be used for creating hostile NPCs. Then, the NPCs will have a similar the abilities and action patterns adaptive to game players.

In Section 2, the related research is introduced and in Section 3, the player modeling using D-FSM and its application to NPCs is described. The experiment result is examined in Section 4. Finally, Section 5 concludes this paper and suggests future works.

2 NPC(Non-player Characters) and FSM(Finite State Machine)

2.1 Decision Tree in Games

Decision tree method, which is a technique widely used for data classification in data mining illustrates the patterns in the data in a tree structure by analyzing the given data. The reason why it is frequently used is that its' output is in a simple formal and can be easily understood by users. It has been applied to 'Black & White' and has a big favor from users. In the game, the player can give positive or negative feedbacks by caressing or beating the creature. A decision tree is generated from the feedbacks. Sometimes a creature checks if given objects will release hunger using the decision tree and sets the degree of aggressiveness. For example, if the player treats a creature with violence rather than praise more often, the creature gets violent more and more.2.1 FSMs for NPCs

2.2 FSM (Finite State Machine)

The state machine or finite state machine, which is one of widely used software designing patterns, is a machine which has finite numbered states [4]. An FSM is defined with the finite collection of states and transitions, between states. In applications, each state of an FSM may be attached with a specific action. NPCs conduct the actions attached to the present event state as their tasks. When a certain occurs, a state transits into another state will follow. FSMs are most widely used technique in games as it has a simple structure. FSMs are sometimes used mixed with other artificial intelligence techniques. Figure 1 illustrates an FSM.

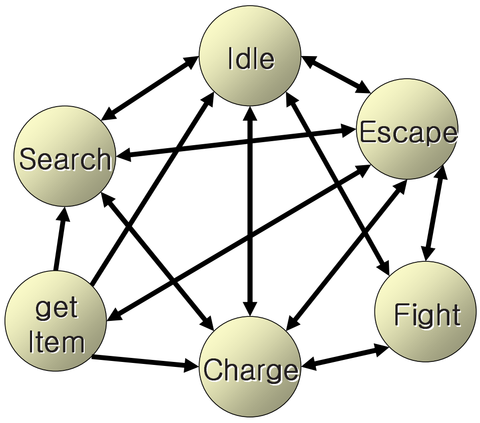


Fig. 1. 6 states and transition relation

Here it has 6 states and each state can be transited to the other states. For example, if the NPC in a state of “Idle” is attacked by the player, its’ state will change into a state of “Fight” and if the NPC is injured, its’ state will change into a state of “Escape”.

2.3 Elementary NPCs and Improved NPCs

NPCs are agents that respond to game players using various elements. They can be merchants in stores, guides showing ways, teammates with game players, monsters, environmental elements such as the climate and geographic varieties and the tool items which game players use. The behavior patterns of these NPCs are directly set in source code or script, so never change during game playing. NPCs created by an improved method, usually show various responses according to the users` tendencies.

Table 1 and 2 illustrate the functions of NPCs in usual games and the examples of games to which elementary and improved NPCs are applied.

For elementary NPCs, the action patterns are defined in source codes directly in most cases. This causes the disadvantage that improving NPCs is difficult and that NPCs have static action patterns. Adaptable NPCs adjust their ability and play with

Table 1. Elementary NPCs

Description	▪ Game players` abilities are not reflected to by NPCs. ▪ NPCs show the same response to the same situation. ▪ NPCs` game patterns can be grasped. ▪ NPCs show the same static action patterns according to situations.
Applied Examples	Space Invaders, Pac-Man , Donkey Kong and the like

Table 2. Improved NPCs

	Ability-Adaptable NPCs	Behavior-Adaptable NPCs
Description	▪ NPCs` abilities vary accord ing to game players' abilitie s. (hitting accuracy rate, sp eed, energy and the like) ▪ NPCs control the level of g ame difficulty dynamically.	▪ NPCs` action patterns vary accor ding to the tendencies and abilitie s of game players. ▪ NPCs are dynamically developed in games. ▪ NPCs contribute to enhancing fun by raising the realistic feeling of games.
Applied Ex amples	FIFA, Call of Duty, Meda l of Honor and the like	Black & White, The Sims and t he like

the new ability in the next games. Ability-adaptable NPCs change only numeric parameters, such as the level of difficult, hit ratio, power, etc. and do not change the behavior patterns. On the contrary, behavior-adaptable NPCs show dynamic response to the game players by analyzing the game player's gaming data. Behavior-adaptable NPCs change behavior patterns or even take the creative and more developed actions which were not designed in the initial phase of game production. The D-FSM suggested in this paper can be utilized as a fundamental technique for providing behavior- adaptable NPCs through adjusting the initially defined FSM of NPCs using the gaming data of game players.

3 Dynamic-FSM Method Using Player's Gaming Data

This paper proposes the D-FSM which is the method to alter transition rules of FSMs defined in the initial phase using the gaming data of players collected in games. To create NPCs which can adapt to game players, the gaming data of players should be collected first. The modeling on game players are conducted by analyzing the collected data. The model of game players will be created in a form of FSMs. The FSMs will be applied to NPCs to change the behavior patterns of NPCs. The diagram in Figure 2 illustrates the D-FSM process.

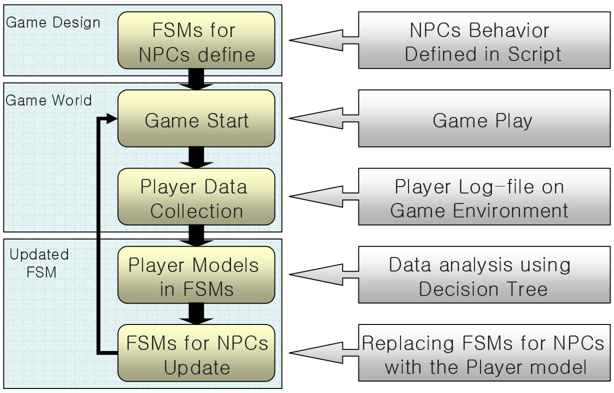

Game Design
FSMs for NPCs define
Game World
Game Start
Player Data Collection
Updated FSM
Player Models in FSMs
FSMs for NPCs Update
NPCs Behavior Defined in Script
Game Play
Player Log-file on Game Environment
Data analysis using Decision Tree
Replacing FSMs for NPCs with the Player model

Fig. 2. Work-flow for D-FSM

3.1 FSMs for NPCs

Figure 1 shows the state transition of NPCs which is generally used in FPS (First Person Shooter) or MMORPG (Massively Multi-player Online Role Playing Game).

Table 3. State transition rules

S_{from}	PH	NH	HD	D	SD	PD	S_{to}
Search	H	H	0	O	1	1	Search
Search	M	H	0	I	-1	0	Charge
Search	M	M	1	I	1	-1	Idle
Search	H	L	1	I	0	0	Escape
Charge	H	H	1	I	0	-1	Charge
Charge	H	M	1	I	1	-1	Idle
Charge	L	L	1	I	1	-1	Fight
Charge	L	M	0	O	1	0	Search
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
getItem	M	M	0	O	0	0	Idle
getItem	M	H	0	O	1	1	Search

NPCs take the actions defined in its' current state. The transitions between states are usually defined with situation variables.

The representative situation variables are the body strength of PC (Player Character) and NPC, and the difference of body strength, attack power, moving speed and distance between PC and NPC. As additional variables, the items which characters own can be considered. Various other abilities of characters and the existence of comrade characters in the vicinity also may be counted.

After the states for NPCs are defined and the variables which are used for state transitions are identified, the state transition rules as shown in Table 4 should be generated.

Table 4. Variables used for defining transitions and their description

Variable	Attribute values	Description on Attribute values
PC Health Point (**PH**)	Low(L), Mid(M), High(H)	0~40=L, 41~70=M, 71~100=H
NPC Health Point (**NH**)	Low(L), Mid(M), High(H)	0~40=L, 41~70=M, 71~100=H
Health Difference of PC and NPC (**HD**)	0 ,1	PH < NH = 0, PH > NH = 1,
Distance of PS and NPC (**D**)	Near(N), Inside(I), Outside(O)	N : attack possible distance. I : inside of view sight. O : outside of view sight.
Speed Difference of PC and NPC (**SD**)	-1,0,1	If PC is faster than NPC, 1. If NPC is faster than NPC, -1. If the speed is the same, 0.
Power Difference of PC and NPC (**PD**)	-1,0,1	If PC is stronger than NPC, 1. If NPC is stronger than NPC, -1. If the power is the same, 0.

The transitions in Table 3 are defined using the values of 6 variables shown in Table 4. For example if the NPC is currently in the state of "Search", the health of PC and NPC are 80 and 39 respective, PC is in the view sight and the moving speed and the attack power of PC and NPC are the same, the next state of NPC will be "Escape", which is what the fourth rule says. Defining state transitions by scripts is one of most frequently used techniques because it is easy in the maintenance and expansion of NPCs.

3.2 Modeling Players with FSMs

This section will describe how to model players' palying patterns with FSMs which will be applied to NPCs. Players' playing data can be modeled in various ways. The reason for modeling with FSMs is to make the model easily adapted for updating NPCs' FSMs. If players' models are applied to NPCs, an NPC will play intelligently or in a way similar to how players' play in the game.

In order to model players' playing patterns with FSM, it is assumed that players also have a hidden FSM and choose actions based on the FSM. However, it is not easy to infer what states the player has it is also assume that the player's FSM has the same states as NPCs'. The current state of the player is inferred from observations and some heuristics.

NPCs in FPS games usually have six states shown as in Figure 1. The description of states and heuristics for inferring the player's current state are as follows:

Idle – In this state is that the game player takes no action. He may wait for the comrade or the enemy, or takes a rest on a specific location. If situation variables do not change for a certain time interval (5~10 sec.), the player is regarded as in "Idle" state.

Charge - This is the state to approach toward the enemies. If the player is in enemy's view and approaching to the enemy, the player is regarded as in "Charge" state.

Fight - This is the state to combat with enemies which are in attackable distance. If an enemy is within attackable distance and attack keys are inputted, the player is regarded as in “Fight” state.

Escape - This is the state to retreat from enemies when game players feel disadvantageous in combat or when game players were in low body strength condition. If the player is in enemy’s view and the distance between them is increasing, the player is regarded as in “Escape” state.

getItem - This is the state where game players search tool items near from him to recover the body strength. If the player gets near to items and the player’s current state is “Escape”. The player is regarded as in “getItem” state.

Search - This is the state to move around to search enemies. So, if the player is out of enemy’s sight and moves around, the player is regarded as in “Search” state.

When a game player’s action and the values of situation variables are consistent with the heuristic condition of a state, the game player is regarded as in that state. For example, if the enemy is out of the player's sight and the player is moving around, it is concluded that the player is currently "Search" state. If the enemy gets into the player's sight and the player gets closer to the enemy, the player is regarded charging the enemy, i.e., in "Charge" state.

The next step is acquiring the player's transition rules between states. For acquiring transition rules, the values of situation variables are recorded whenever the player's state changes. That is, the values is collected in a form of (S_{from}, PH, NH, HD, D, DS, PD, S_{to}). The collected data is pre-processed to remove duplications and decision tree method is applied to find out transition rules in the data.

The result of decision tree learning is expressed in a tree structure which can be described in a table format like tables. The rules newly created will replace the NPC’s transition rules. That is, it is used to make NPCs intelligent or human-like. The newly created NPC rules are not static rules. These rules can be created from the game player’s gaming data whenever new rules need.

4 Experiments

The experiment was conducted using the Half-Life by Valve Software[7] game and Jeffrey’s HPB Bot[8], where a PC and NPC combat were used under the method suggested in this paper. This game initially sets the states and transition rules of the NPC by reading the scripts and collects the player’s gaming data while the game is going on. The player can move forward, backward, left and right, and attacks the NPC.

The PC and the NPC initially have the same moving speed and power. The power given to the PC and the NPC in the initial phase is 100. If the values of situation variables do not change for more than 5 seconds, it is defined that the player is in “Idle”.

During game play the values of the body strength of the PC and the NPC, the difference of the body strengths, the distance between the PC and the NPC, and the difference of the moving speeds and the attack powers are collected, whenever the state of the game player changes.

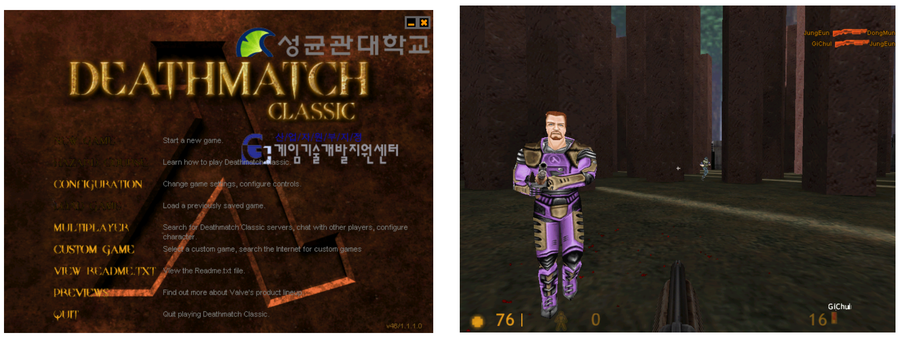
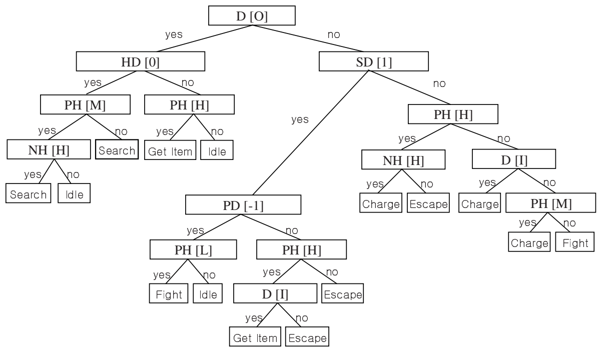

성균관대학교
DEATHMATCH
CLASSIC
게임기술개발지원센터
CONFIGURATION
MULTIPLAYER
CUSTOM GAME
VIEW README.TXT
PREVIEWS
QUIT
Start a new game.
Learn how to play Deathmatch Classic.
Change game settings, configure controls.
Load a previously saved game.
Search for Deathmatch Classic servers, chat with other players, configure character.
Select a custom game, search the Internet for custom games
View the Readme.txt file.
Find out more about Valve's product lineup.
Quit playing Deathmatch Classic.
JungEun
DongMun
GiChul
JungEun
GiChul
76
0
16

Fig. 3. DeathMatch Class Game in MOD of Half-life and Game Screenshot

The game recognizes the state of the PC on the basis of the heuristics previously described. After one round of the game, the NPC learns the collected data using decision tree method and updates its' transition rules. Figure 4 illustrates the initial transition rules in Table 3. Those are represented in a tree format. Figure 5 represents the newly learned transition rules from the player's gaming data. It is noted that the newly learned transition rules are quite different from the initial one. It has more complex rules for "Charge" than the original but simpler ones for "Search". The NPC takes actions more intelligently than with the initial ones.



* Meaning of Node : Variable [Attribute value]
* Learning method : Decision Tree (ID3).
* Training data : Initial transition rule in Table 4.

Fig. 4. Initial transition rules in a tree format

In experiment, 15 players play the game with the initial rules for NPC's, and then play again the game with the new rules learned from his previous game play data. We measure how long each player play the game and many times each player die during play. We also ask a few questions to the players: the degree of difficulty and the degree of enjoyment. Table 5 shows the result. The players answer that the game with learned rules are more interesting and difficult to win. The average playing time is increased from 217.3 to 251.0.

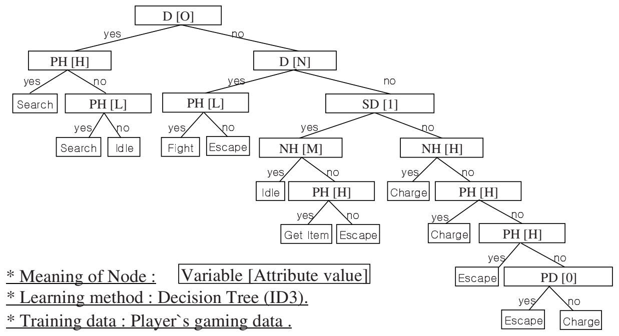


Fig. 5. A newly learned transition rules from the player's data

Table 5. Experiment result using D-FSM

Players	NPC of initial transition rules				NPC of learned transition rules			
	Game Data		Satisfaction (0~100)		Game Data		Satisfaction (0~100)	
	Game progress time(sec.)	Death number of times	Degree of difficulty	Degree of enjoyment	Game progress time(sec.)	Death number of times	Degree of difficulty	Degree of enjoyment
Player 1	184	5	55	65	272	7	80	85
Player 2	130	5	60	70	172	5	90	90
Player 3	205	4	65	60	402	4	100	100
Player 4	271	9	55	45	221	5	70	98
Player 5	328	7	60	50	196	6	60	95
Player 6	253	8	70	60	265	9	80	95
Player 7	266	7	50	70	287	6	90	85
Player 8	204	6	40	45	131	2	70	90
Player 9	200	5	50	55	261	6	40	80
Player 10	133	2	60	50	205	3	80	85
Player 11	182	4	70	65	184	7	90	90
Player 12	234	5	55	50	290	5	70	70
Player 13	178	6	65	30	221	7	80	60
Player 14	278	5	40	50	324	6	80	80
Player 15	213	4	30	70	334	7	90	85
Ave.	217.3	5.5	55.0	55.7	251.0	5.7	78.0	85.9

* Mission of this game is killing NPCs 10 times.
* Game progress time: It is cost time to kill NPCs 10 times.
* Death number of times: It is number of times that player dies during mission achievement.
* Degree of difficulty: If it is near to 0, easy. If it is near to 100, difficulty.
* Degree of enjoyment: If it is near to 0, boring. If it is near to 100, interesting.

5 Concluding Remarks

NPCs provided in games are usually have static action patterns. These static actions of NPCs are one of factors which degrades the fun of games and makes the life-cycle of games short. This study is on an approach which can provide creative and diversified NPCs by updating FSMs through collecting and analyzing the gaming data of players. This method can be used in other the process of game development, such as designing initial FSMs for NPCs, creating partner NPCs of players which cooperate with players and creating hostile NPCs in games. To generate a more compact rule set pruning method which reduces the number of rules will be applied as a future work.

References

[1] Brajnik, G., Guida, G., Tasso, C.: User modeling in expert man-machine interfaces: a case study in intelligent information retrieval. IEEE Trans. SMC 20(1), 166–185 (1990)
[2] Cha, H.J., Kim, Y.S., Lee, J.H., Yoon, T.B.: An Adaptive Learning System with Learning Style Diagnosis based on Interface Behaviors. In: International Conference on E-learning and Games, Edutainment (2006)
[3] Rabin, S.: AI Game Programming Wisdom 2, Charles River Media (2002)
[4] Loura, M.D.: Game Programming Gems, Charles River Media (2000)
[5] Laird, J.E.: Using a Computer Game to Develop Advanced AI. Computer IEEE Journal 34(7), 70–75 (2001)
[6] Spronck, P., Kuyper, I.S., Postma, E.: Online Adaptation of Game Opponent AI in Simulation and in Practice. In: Proceedings of the 4th international Conference on Intelligent Games and Simulation, GAME-ON, pp. 93–100 (2003)
[7] Half-Life Game site: http://www.half-life.com
[8] PHB Bot site for Half-Life game: http://hpb-bot.bots-united.com