Using Artificial Neural Networks in NPC Decision-Making Process

Article · November 2013

CITATIONS
0

READS
856

2 authors, including:

Diana Adamatti
Universidade Federal do Rio Grande (FURG)
130 PUBLICATIONS **158** CITATIONS

SEE PROFILE

Using Artificial Neural Networks in NPC Decision-Making Process

Carlos Alberto Barros Cruz Westhead Madsen
Nucleo de Tecnologia da Informaçao
Universidade Federal de Rio Grande - Brasil

Diana F. Adamatti
Centro de Ciencias Computacionais
Universidade Federal de Rio Grande - Brasil
Email: dianaadamatti {at} furg.br

Abstract—**The present paper presents the use of ANN (Artifical Neural Networks) in NPC (Non-Player Character) decision-making process in RPG (Role-Playing Games) electronic games. It mainly focus on the making of decision of attacking or not a certain opponent by analyzing the NPC and its opponent internal state.**

Keywords- games; artifical neural networks; decision-making process; NPC

I. INTRODUCTION

The Artificial Intelligence (AI) applied in electronic games is known as Games AI (Game Artificial Intelligence), and its main purpose is to create a behavior which seems intelligent, due to a scenario with multiple choices [9] [10].

This should be similar to the human-being behavior with personality, making mistakes and be able to provide different levels of difficulty to the employer, in order to add experience and immersion in the game and improve its gameplay [11].

Even though, in the game industry, AI has been used since its beginnings, when it was known as gameplay programming, its full usage is still a challenge, mainly due to the following reasons: development period, learning algorithms testing and performance [12].

Among the several AI techniques applied in games, we highlight the ANN because they have the skill of learning and generalization, through an interactive process with the external environment, mainly because of wide usage in prediction problems [4] [8].

Due to this, the present work aims to contribute in the purpose of presenting a possibility of ANN use in the decision-making process through an interactive process in RPG. In this way, the FSG (FURG Smart Games) framework [2] will be used. It has ANN Black Box type implementation, where all components are abstracted, enabling a quicker implementation and focus on the theme of the present work.

The paper is divided in six sections. In section II there is a discussion on ANN approaching. In section III, the FSG framework is presented. In section IV, the rules of the RPG are detailed in which the AI technique is used. In section V, the experiments performed using the ANN in the NPC decision-making process are presented. Finally, in section VI the conclusions of the present work are presented as well as the further works.

II. ARTIFICIAL NEURAL NETWORKS

The artificial neural networks (ANN) are the main technique of connectionism, an AI line which studies the possibility of simulation of intelligent behavior through mathematical models which seek to resemble the biological neural structures. These are characterized by being distributed processors comprised by a number of simple processing units, known as artificial neurons, which tend to store experimental knowledge, allowing learning and generalization, given an interactive process with the external environment [4] [5].

This AI technique is mainly applied in prediction, classification, categorization and optimization problems, as well as recognition of characters, voice, prediction of time series, process modeling, computer vision and signal processing [6].

Among its main characteristics we may highlight the following [4] [5]:

- Generalization: capacity of learning through examples and thus making generalizations to recognize similar instances to those which have never been presented before;
- Adaptability: possibility of adapting its synaptic weights in order to absorb modifications in the environment. Thus, a network which had been trained to work in certain conditions may be retrained to deal with modifications;
- Contextual information: the knowledge is represented by its own network structure, where each neuron of the network is potentially affected by the activity of all other neurons. Due to this the contextual information is naturally treated;
- Error tolerance: capacity to fulfill its purpose under signals with noises, or even loss of communication in part of the network;
- Self-learning: there is no need of knowledge from a specialist to make decisions. The ANN is based only on historical examples which are provided to it;

- Non-linear modeling: the mapping process of a neural network involves non-linear functions which may cover a bigger limit of the problem complexity.

A. *Learning*

The learning of an ANN takes place through the modification of its synaptic weights in an ordered way through training algorithms. These algorithms could be classified in two distinct paradigms, the supervised one and the non-supervised one [5].

- Supervised: the outputs are known during the training, and a program known as a catalyst monitors the mistake, the difference between the value wished and propagated output, and refeeds the network with the mistake in order to minimize it to an acceptable threshold.
- Non-supervised: in this case, there is no catalyst or refeeds to say whether the learning process was reached. The networks use this type of training are known as self-organizing maps – SOM. They use correlations between the input standards to categorize them according to the self-discovery classes.

B. *Multilayer Perceptron (MLP)*

This work will focus on the multilayer perceptrons feedforward network, totally connected, due to their usage in games [7].

This type of neural network is characterized by having its neurons in multiple layers, normally an input layer, several hidden layers and an output layer. The neuron of each layer is connected with all the neurons of the immediately before and after layers. Thus, given an input, it propagates, according to the integrator and activation function of each neuron, layer by layer, until an output group that produces as a network final response [5]. MLP networks with three layers have the capacity of approaching any non-linear function. In fact, normally the information that a NPC captures from a game environment is non-linear [5], justifying its usage in games.

Finally, the learning in this type of network occurs in a supervised way through the backpropagation algorithm. This is constituted by two basic steps, the network propagation and the error retropropagation. In the first step, a given input sign is propagated throughout whole network and the resulting output is stored. In this phase, the synaptic weights remain unchanged. In the second phase, the network output is compared with values which the catalyst recognizes them as true for the presented input, through the difference between these two groups an error value is created. By making use of this error, the catalyst retropropagates through the network, adjusting the synaptic weights with the purpose of moving it closer to the expected solution. Thus, this happens successively, until the network converges to the output error within an acceptable limit [5].

One question that arises is the possibility that during the training, the MLP finds a minimum place and it do not present the ideal result. However, in the context of electronic games, this is not necessarily a problem, as it is not expected that the NPC has an ideal or perfect behavior. Eventual errors, as long as less predictable, are also interesting.

III. FSG (FURG SMART GAMES)

The FSG framework aims to help the AI techniques incorporation more effective in the development of electronic games, assuming that the decision-making layer of an NPC is an FSM (Finite Sate Machine), a technique widely used in the industry [12] [3].

In order to do it, every time a FSM receives from the environment an event which takes it, in response, passes from its current status to another, the decision-making which makes this transition to happen or not, is ruled by a guard condition, and it may be related to an AI technique. With its final purpose to make the character have a less predictable behavior and it may respond in a more realistic way [2].

In the Figure 1, a classes diagram with the FSG core is presented. In this, three abstract classes are highlighted: FSGCharacter, FSGState and FSGAITechnique. In the FSGCharacter class there is the NPC implementation. It is noticeable the presence of the "currentState" attribute which denotes in which status the machine is, the "transition" method responsible for the transition between status, and the "activity" method, which delegates the responsibility of the character behavior to the method of same name of the object "currentState". In the FSGState, all states in the FSM must be implemented. Finally, the FSGAITechnique class serves as an interface between the AI techniques and the FSM state transitions. Normally, its objects are instantiated in the "guardCondition" method of a given class which inherits FSGstate. In Figure 1, the classes concerning the AI techniques were suppressed.

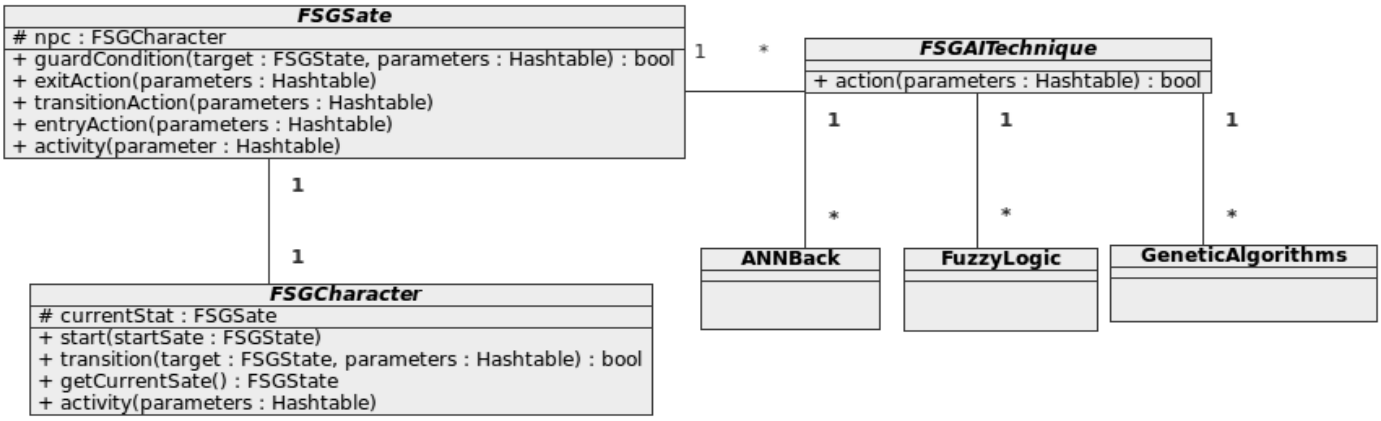

Figure1: FSG Simplified classes diagram

IV. PROPOSED GAME

We have proposed a game where two characters duel in a way similar to the RPG. This game was implemented in Java using the framework JGame [1], for the implementation of 2D animations and the framework FSG [2] for the implementation of the decision-making layer of the character “Guard". It is a very simple game, just to test the “Guard” NPC decision-making process. Figure 2, the game interface is presented, which the following items are highlighted:

Figure 2. Game proposed for the validation of the tool.

1. Character known as “Player", when the game is not on automatic mode, may be controlled by the user through the keyboard directing arrows and the key “A" for the attacks;
2. Character known as “Guard" is the game NPC and its function is to guard a certain area besides defending itself against any other character which may enter its reach;
3. Area where, before starting a game, the player may set the attributes of both characters. It is even possible to define which AI technique will be used by the “Guard" in its decision-making (attack or not). Nowadays, the framework FSG enables the usage of Artificial Neural Networks, Fuzy Logics and Genetic Algorithms;
4. Finally, in this interface there is the possibility of automatically simulating a certain number of rounds, in which the two characters are autonomous and their attributes are defined randomly. At the end of each duel, the information concerning the initial attributes of the characters, as well as who the winner was, is stored in a file for later training of the neural networks.

Each game characters has the following attributes:

- Energy: it represents its vital energy, with restricted values at the interval [0,100]. When a character’s energy equals zero, this is considered as defeated and the game is finished;
- Experience: this attribute defines the character’s experience in battle, with values in the interval [0,100]. This value establishes the possibility of a carried out attack to be successful or not. Thus, in case a character has 20 points of experience, he will have a 20\% probability of causing some harm to its opponent;
- Attack power: this attribute sets, in case of a successful attack, the harm maximum value the opponent may suffer, with the values in the interval [0,100];
- Defense power: it defines how much harm the character can stand before its vital energy is affected, with the chance of receiving values in the interval [0,100];
- Reach: this attribute sets the distance around which can be perceived by the character concerning the environment, with values in the interval [150,250];
- Speed: this attribute defines the speed in which the character moves around the environment, with values in the interval [0,10].

The harm that a certain successful attack causes to the opponent is defined by the Equation (1). In this, it can be noticed the direct dependence in relation to experience, vital energy and attack power. Due to this, the harm will only be equal to the attack power when the character has its energy and experience in their maximum values.

$$Harm = (\frac{experience}{100}).(\frac{vital\_energy}{100}).(attack\_power) \quad (1)$$

Briefly, a confrontation may be defined through the following steps:

1. The characters move around the environment until an opponent is in its reach;
2. Once an opponent is within reach, an attack is carried out;
3. If, according to the character’s level of experience, the attack is successful, then the harm is calculated according to the Equation \ref{eq:dano};
4. If the harm inflicted is smaller than the opponent’s defense power, this attribute is decreased;
5. If the harm inflicted is bigger or equal to the opponent’s defense power, its defense item (for example, the shield) is destroyed and its vital energy is vulnerable for a next attack;
6. If the opponent’s defense power is zero, then the amount is deducted from the opponent’s vital energy;

7. At the end of a successful attack, the offender's experience is added a point;
8. The offender passes its turn to the opponent, if it is within his reach, so it can carry out the next attack;
9. This cycle is repeated until one of the two is defeated or is out of reach.

The NPC "Guard"' behavior ruled by the Finite State Machine (FSM) illustrated in Figure 3.

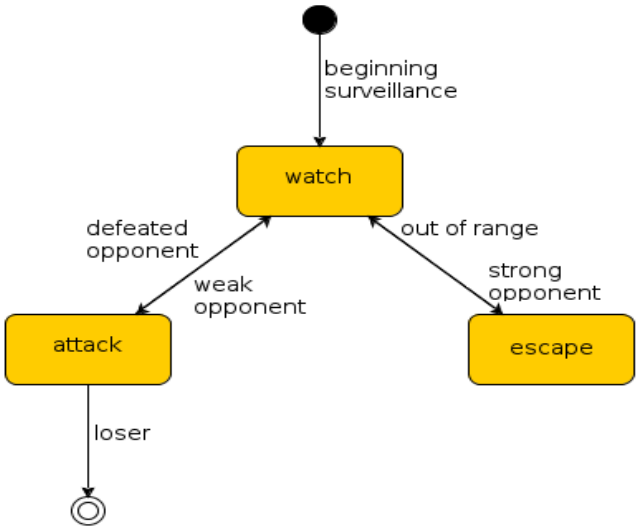

Figure 3. FSM responsible for the NPC "Guard"

According to the FSM presented in Figure 3, the Guard initially assumes the "Survey" state, where the NPC moves around the area where it keeps guard and if an opponent is noticed, according to its internal attributes and the enemy's, should decide either to attack or escape.

In this work, this decision-making process is carried out, associating to an ANN the function of "Weak opponent" transition guard, leading to the "Attack" state, and also a function of "Strong opponent" transition guard, leading to the "Escape" state. In the "Attack" state, there is a confrontation between the two characters and, if it wins, the "Guard" returns to the "Survey" state. If not, the process is finished. Once taking the "Escape" state, the NPC tries to be away from the enemy's reach. If successful, it returns to the "Survey" state.

V. EXPERIMENTS AND RESULTS

For the training of the ANN used in the game, two distinct files were created, each one resulting from 500 rounds carried out automatically. In this automatic mode, the NPC Guard always makes the decision of attacking when there is an enemy within its reach. In each line of these files, the initial attributes of each character were registered, as well as who was the winner of each duel.

The first automatic simulation resulted in 243 (48,6%) victories of the Guard and 257 (51,4%) of the Player, in which a description of their attribute values is presented in Tables I and II, respectively.

TABLE I: Guard Attributes In The First Simulation.

Attribute	Average	Standard Deviation
Energy	51.56	27.88
Experience	53.01	27.92
Attack Power	49.07	27.77
Defense Power	48.62	29.29
Reach	200.91	29.94
Speed	4.61	2.91

TABLE II: Player Attributes in the first simulation.

Attribute	Average	Standard Deviation
Energy	51.56	27.88
Experience	53.01	27.92
Attack Power	49.07	27.77
Defense Power	48.62	29.29
Reach	200.91	29.94
Speed	4.61	2.91

The second automatic simulation resulted in 253 (50,6%) victories of the Guard and 247 (49,4%) victories of the Player, in which a description of their attribute values is presented in Tables III and IV, respectively.

TABLE III: Guard Attributes in the second simulation.

Attribute	Average	Standard Deviation
Energy	50.89	29.39
Experience	52.76	27.49
Attack Power	49.25	28.43
Defense Power	50.04	28.39
Reach	202.27	29.91
Speed	5.13	2.82

TABLE IV: Player Attributes in the second simulation.

Attribute	Average	Standard Deviation
Energy	51.43	29.43
Experience	52.98	28.74
Attack Power	49.48	29.26
Defense Power	48.38	29.80
Reach	202.06	29.20
Speed	4.77	2.79

A. Artificial Neural Network

In this work an ANN was implemented with the following configuration:

- Twelve neurons in the input layer, where the first six ones are destined to receive the Player attributes and the five last, the Guard attributes;
- Twenty-four neurons in the hidden layer;
- Two neurons in the output layer. If the Player wins, the first neuron must present the value one and the second zero, and if the Guard wins these values are inverted;
- Learning rate in 0.4;
- Momentum in 0.2;
- Sigmoid activation function (standard in framework).

This network training was designed with the following parameters:

- Acceptable error in the output layer: 0.3%;
- Maximum number of periods: 500,000;
- Presentation of each standard for the network for five times;
- 10% of the training data was separated for cross-validation;
- 100 networks were trained, and, from these, the one with less output error was extracted.

In this training process, 450 registrations were used for the training and 50 for the network validation. Besides that, there was the need of data normalization, with all between -0.5 and +0.5.

Once the best network had been tested, it was propagated over all the data of the simulation files, with the purpose of checking if the decision made was right. This decision-making process developed in the following way: if the first output neuron is higher than the second, the Guard must escape, otherwise it must attack.

By processing the file referring to the first simulation, which served as training for the network, the following results were reached:

- Decision of attacking in confrontations in which the Guard was the winner: 240 of 243 victories, in a total of 98.71%;
- Decision of escaping in confrontations in which the Guard was defeated: 255 of 257 victories, in a total of 99.22%.

When the resulting file was processed for the second simulation, to which the network had never been presented, the following results were reached:

- Right decisions, including the attacks and the escapes: 424 out of 500, in a total of 84.8%;
- Decision of attacking in confrontations in which the Guard was the winner: 210 of 253 victories, in a total of 83,01%;
- Decision of escaping in confrontations in which the Guard was defeated: 214 of 247 victories, in a total of 86,63%.

VI. CONCLUSIONS AND FUTURE STUDIES

The AI techniques use in NPC decision-making layer, mainly in FSM, enables the characters to have more realistic behaviors. Due to the prediction capacity the ANN acquire for their training, its use, in the context, is providential. This is proved by the rate of right decisions that this technique presented in the proposed experiments.

An important issue, which was not the focus of the present work, concerns the moment in which the ANN is trained. The training presented took place off line, ie, before the game was executed. In future studies, the intention is to study the possibility of training the network during the game execution so that besides learning its rules, the network can adapt to the player's strategies.

REFERENCES

[1] B. Schooten, JGame - a Java/Flash game engine for 2D games. http://www.13thmonkey.org/boris/jgame/ (2013).

[2] C. A. B. C. W. Madsen, Ambiente de desenvolvimento de jogos com reuso de software e inteligencia artificial. Programa de Pos-Graduacao em Modelagem Computacional, FURG, Rio Grande, Brasil (2012).

[3] C. A. B. C. W. Madsen, D. F. Adamatti., G. Lucca, G. Daniel, FURG Smart Games: a Proposal for an Environment to Game Development With Software Reuse and Artificial Intelligence. The Fourth International Conference on Networked Digital technologies. p. 369-381 (2012)

[4] S. O. Rezende, Sistemas Inteligentes Fundamentos e Aplicacoes. 1st ed. 370 pp. ISBN 8520416837. Edit. Manole Ltda, Barueri - SP, Brazil (2005)

[5] S. Haykin, Redes Neurais Principios e Praticas. 2nd ed. 900 pp. ISBN 0132733501. Edit. Bookman, Higienopoliss - SP, Brazil (2001)

[6] S. Russel, P. Norwig, Inteligencia Artificial. 2nd ed. 1056 pp. ISBN 8535211772. Edit. Campus, Rio de Janeiro - RJ, Brasil (2004)

[7] P. Sweetser, J. Wiles, Current AI in games: a review. Australian Journal of Intelligent Information Processing Systems, vol. 8, nr. 1, p. 24-42 (2002)

[8] K. Chellapilla, B. D. Fogel, Evolution, neural networks, games, and intelligence., Proceedings of the IEEE, vol. 87, nr. 9, p. 1471-1496, ISSN 0018-9219 (1999)

[9] R. T. Santana, IA Em Jogos a Busca Competitiva entre Homem e a Maquina, Faculdade de Tecnologia de Praia Grande.,Praia Grande, Brasil (2006)

[10] J. D. Funge,Artificial Intelligence for Computer Games: An Introduction., AK Peters/CRC Press, ISBN 9781568812083 (2004)

[11] B. Schwab, AI Game Engine Programming., Hingham: Charles River Media, ISBN 1584503440 (2004)

[12] D. Bourg, G. Seemann, AI for Game Developers, OReilly ISBN 0596005555 (2004)