DEPLOYING FUZZY LOGIC IN A BOXING GAME

Hamid Reza Nasrinpour, Siavash Malektaji, Mahdi Aliyari Shoorehdeli and Mohammad Teshnehlab
Department of Electrical and Computer Engineering
K. N. Toosi University of Technology
Seyedkhandan, Tehran, Iran
nasrinpour, siavashmalektaji@ee.kntu.ac.ir
m_aliyari, teshnehlab@eetd.kntu.ac.ir

KEYWORDS
Fuzzy Logic, Computer Game, Rule-Based System, Boxing

ABSTRACT

Nowadays computer games have become a billion dollar industry. One of the important factors in success of a game is its similarity to the real world. As a result, many AI approaches have been exploited to make game characters more believable and natural. One of these approaches which has received great attention is Fuzzy Logic. In this paper a Fuzzy Rule-Based System is employed in a fighting game to reach higher levels of realism. Furthermore, behavior of two fighter bots, one based on the proposed Fuzzy logic and the other one based on a scripted AI, have been compared. It is observed that the results of the proposed method have less behavioral repetition than the scripted AI, which boosts human players' enjoyment during the game.

INTRODUCTION

The academic definition of Artificial Intelligence (AI) states that AI is creating machines which can sensibly think and act as humans (Russell and Norvig 1995). This classic AI is usually concerned with the optimum solution to a problem and discusses how this solution can be found. Game AI is a code or technique which a computer uses to control Non-Player Characters (NPCs). It does not care how a computer makes a decision or thinks about the problem, it might use either a decision tree or a huge database, the point is just its reaction.

Game balance has been one of the considerable topics in Game AI. If a computer opponent always just followed similar patterns, or played too easily or strictly, the game would be annoying. Furthermore, the aim is not to defeat players, but rather to keep them amused. In fact, none of the opponents should be unbeatable.

In Game AI, natural game laws should be followed and cheating should be prevented as far as possible. Cheating AI is a term which refers to a situation where the AI has more information and advantages than the players. This is often implemented in games to increase the abilities of a machine and it might be acceptable if the player is not apprised of it. Some examples of cheating AI can be found in (Scott 2002). As a matter of fact, applying these cheats represents weaknesses and limitations of AI against human knowledge. Therefore it would be better to look for the methods which represent human knowledge, like Fuzzy rule-based systems, as they are a widespread form of Fuzzy logic (Zadeh 1965).

Fuzzy logic can easily rate any input based upon importance. Additionally it is really suitable to do multiple operations at once (Doss 2006). Human knowledge and the ways in which humans think and infer can be directly put into a game by Fuzzy rule-based systems. For instance, in a Fighting Game (Rollings and Ernest 2006), a game developer can take advantage of Fuzzy logic to reach higher levels of realism and human-level AI in behavior of the opponent's character. This makes the human players feel that they are playing against another human, not an omniscient computer, which knows about all of the angles and distances between itself and its opponent and consequently, makes no mistake unless it deliberately decides to make the game easier for the human opponent.

In recent years, we have witnessed many applications of Fuzzy techniques in the domain of games. The game *S.W.A.T. 2* has employed Fuzzy logic as an instance of action games (Johnson and Wiles 2001). In the genre of Real Time Strategy (RTS) games, *Civilization: Call to Power* uses Fuzzy State Machines (FuSM) as well (Johnson and Wiles 2001). The great performance of Fuzzy methods in 2009 simulated car racing championship can be found in (Loiacono et al. 2010). Another Fuzzy controller for a car racing championship is discussed in (Perez et al. 2009), and also a Fuzzy-based architecture has been tested in *The Open Car Racing Simulator (TORCS)* in (Onieva et al. 2010). There is a Fuzzy Q-learning method which has been implemented in the game of Pac-Man (DeLooze and Viner 2009). An agent-based Fuzzy system has been applied in the *Battle City* game into the bargain (Li et al. 2004).

In this paper, a method without cheating has been introduced for applying Fuzzy logic in a fighting game. In order to avoid cheating and make a natural and human-level AI, first the game engine was implemented independently. Then the game engine would give control of the characters to a human player or an AI in the same way. Moreover, the rules which the Fuzzy AI fires are the Fuzzy rules which can be followed easily by a human and do not have any complicated computational overhead.

BACKGROUND

Boxing is an old Atari 2600 video game released by Activision group in 1980. *Boxing* shows a top-down view of two boxers which can punch their opponent, as shown in Figure 1. The choice between punching hands (right or left) is made automatically and the human player just presses the hit button. Two boxers are always in front of each other and they can only move up, down, forward and backward. So they cannot rotate. The game finishes after two minutes or when one boxer knockouts the other one by giving him 100 punches.

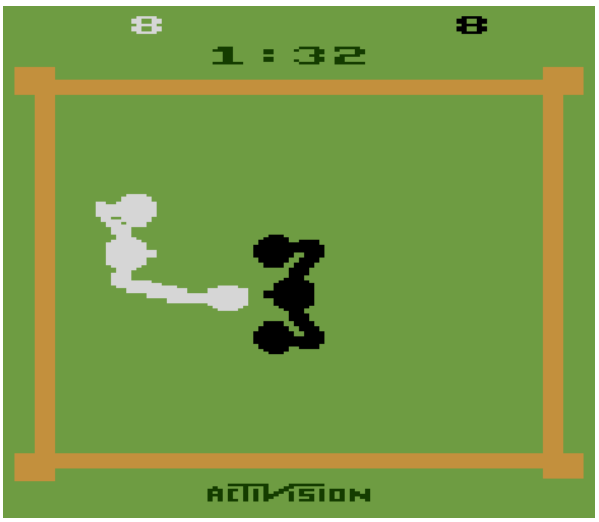

Figure 1: Original Atari Boxing Game

Clever Boxer, discussed in this paper, is an extension of the original *Boxing* game where boxers can rotate and go behind the opponent and the punching hand selection is made manually. So the human player must press the proper button. The boxers can make fast movements in normal directions (forward, backward, left and right) and if they try to punch while moving fast, their punch on the opponent will be more powerful. In addition, the boxer must have enough stamina to punch or to move fast.

GAME ENGINE

The game engine has been designed based on physical rules in order to be able to be implemented on boxer agent bots. Hence the game could be considered as an agent-based simulation software of a real boxer agent's behavior.

Design

This section covers some important rules of the game. The boxer agent, whose behavior is implemented by a Finite State Machine (FSM), has three different types of commandable states which can directly be given by a human or AI player, which are shown in Table 1. If a boxer is commanded to punch, it will stand still and throw a punch. But the commands of rotation and the commands of direction can be given and run simultaneously. The act of throwing a punch, implemented by a FSM, takes place in 4 states. In fact, in each state the hand of the boxer reaches out slightly in a way that in the 4th state the hand is completely stretched. When a punch hits the opponent, it will be effective, if and only if the hand is in the 3rd or 4th state. Besides, the powers of punches in 3rd and 4th states are different from each other.

There are some other states that are not directly commandable and a boxer could be in. An expected state of these non-commandable states is the *Overshooting* state in which the boxer is punched in its back or stomach. The human player or AI does not have any control over the boxer in *Overshooting* states. The complete list of non-commandable states with exact definitions, which implicitly explain the rules of the game, is shown in Table 2.

Table 1: Commandable States of Boxer Agent

State Type	State Name	
Direction States	1.Fix 2.GoingForward 3.GoingBackward 4.GoingRight 5.GoingLeft	6.GoingForwardFast 7.GoingBackwardFast 8.GoingRightFast 9.GoingLeftFast
Punching States	1.PunchRight 2.PunchLeft	3.NoPunching
Rotation States	1.RotateRight 2.RotateLeft	3.NoRotation

Table 2: Non-Commandable States of Boxer Agent

Name	Description
BeingPushed	The boxer is being hustled by the other one. It happens when a boxer is moving faster than the other one.
BoxerContact	Two boxers have a collision with each other.
CircularOvershooting	The boxer is punched in its eye, and then it rotates up to 45 degrees while overshooting.
DoubleOvershooting	If the back of a boxer come into contact with the ring border while overshooting, it will be in this state.
FastPunching	The boxer punches while fast moving. This punch is more powerful than a normal punch.
Overshooting	The boxer is punched in its back or stomach and the human player or AI cannot control it.
Punching	The boxer punches while normal moving.
RingAccelaration	The boxer pushes its back toward the ring border to go forward faster. If a boxer punches in this state, it will go to the *FastPunching* state.
RingContact	The body of the boxer has touched the ring border

As it was mentioned earlier, a boxer will be unable to punch or to move fast if its stamina is lower than a specific level. Also it will not be able to punch if it receives too many punches in its hand. In other words, every hand can endure up to a specific number of hits; otherwise the hand will lose its ability.

Implementation

The game framework, written in Microsoft Visual C#, provides a control loop driven by an external timer to handle animations and collisions. The framework also gives AI an opportunity to perform its own processing, while human players can asynchronously command their boxers.

If AI were reliant on the speed and precision of its punches rather than its playing strategy, it could be seen as cheating because it is impossible for a human player to replicate the computer's effort as fast or as easily as the computer. Therefore the game engine applies two different types of delay to AI in order to make it use more realistic game strategies and act like a human; one is the information delay, which is the delay of receiving data of the environment and the opponent's character from the game engine. The other one is the delay in execution after which a command from the AI is issued. The game engine informs AI of the nature of these delays.

AI

Two different AIs are implemented to control the boxer during the game. One is based on scripted AI and the other

one is based on Fuzzy logic. In this section, these two implementations are discussed.

Scripted AI

Scripting (Bourg and Seeman 2004) is currently the most common means of control in Game AI. Most developers resort to scripts to implement Game AI for complex games where the number of choices at each turn varies from hundreds to even thousands. Some advantages of scripting are being understandable, easy to implement and easily extendable (Tozour 2002).
In this method, some scripts have been written for different situations which might happen during the game. For instance, when a boxer is too close to the opponent in a way that its punches are not effective, it pushes forward the opponent in a fast movement, or when its hand does not have enough stamina for punching, it keeps itself at a safe distance from the opponent, where the opponent's punches would not be efficient and looks forward to the timeout.
Furthermore, some parameters have been defined based on the AI character's and the opponent's health, stamina and abilities of their hands. A selection of these parameters has been used in some scripts. A simple example of the parameters' effects on the scripts is that the scripted AI will play its conservative scripts if the AI boxer's health is less than half of its opponent's health.

Fuzzy Rule-Based AI

Figure 2 provides an overview of the architecture of the game engine and its relation to the Fuzzy rule-based AI. It is based on a classical three-layer hierarchy: perception, control and actuation layer.

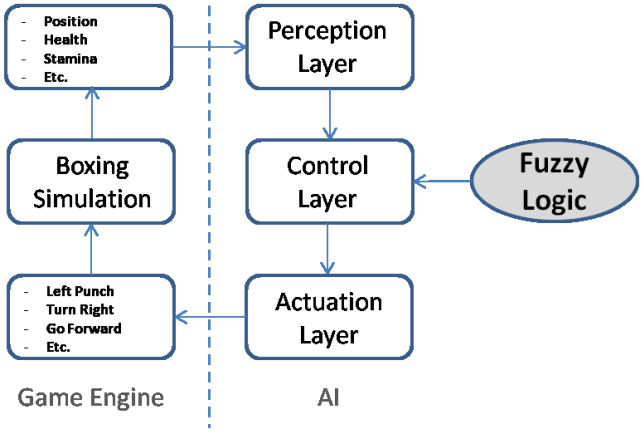

Figure 2: Overview of the system architecture

The perception layer, which receives the position, health, stamina and hand abilities of two boxers from the game engine, first of all, anticipates the current situation of the boxers based on the information delay, the execution delay and its own boxing ring simulator. After that, calculation of the Fuzzy system inputs like *Alpha*, *Distance* and *Danger* is performed.
Alpha as shown in Figure 3 represents the degree between the perpendicular line to the body of the AI boxer, and the link from the central point of a boxer to the other one. *Distance* represents the space between the boxers. Since the AI boxer tries to escape from the corners of the ring, the danger of a position should be estimated independently of the opponent's position. Hence *Danger* represents how dangerous the current position of the AI boxer is.
In the control layer, first of all, by using a script, the AI checks whether its boxer can throw a punch at the opponent or it should change its position. If its punch does not hit the opponent, the AI will call its Fuzzy system to navigate the boxer toward the best position. The main idea of Fuzzy system is dividing the problem into two sub-problems including: 1) Finding the opponent and moving toward it. 2) Adjusting the exact distance and angle for an efficient punching to the opponent and evading the opponent's punches. Herein, a Fuzzy subsystem copes with each sub-problem and a 3rd Fuzzy subsystem supervises the efficacy of each of those two subsystems.
As a direct solution to the sub-problems mentioned above, the Fuzzy system includes three Fuzzy subsystems: 1) *Near Fuzzy System*, which specifies the importance of *Distance*, the AI boxer's stamina and the distance of the AI boxer's hand to its opponent and the opponent's hand to the AI boxer. It is most effective when the AI boxer and its opponent are close to each other; 2) *Far Fuzzy System*, which specifies the importance of *Distance*, *Alpha* and *Danger*. It is most effective when the AI boxer and its opponent are far from each other; 3) *Overall Fuzzy System* which specifies the importance of the *Near Fuzzy System* verdict and the *Far Fuzzy System* verdict. It should be noted that all three Fuzzy subsystems operate during the whole decision making loops and all rules are evaluated in parallel.

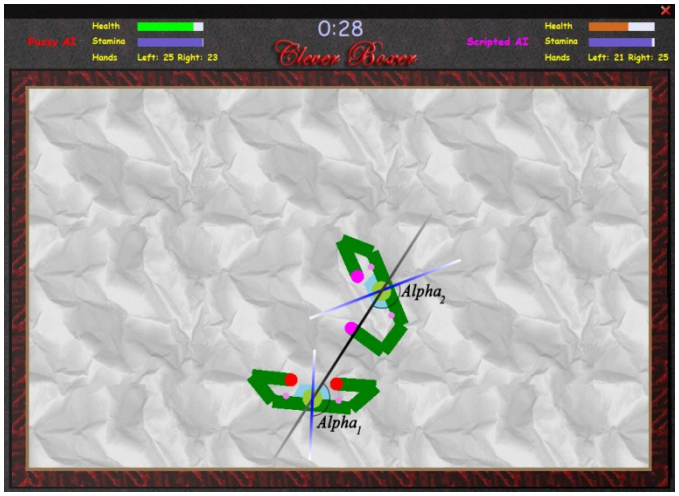

Figure 3: Displaying Alpha in a screenshot of the game

In all of the Fuzzy systems, Takagi-Sugeno Fuzzy model is employed (Takagi and Sugeno 1985). Fuzzy system input variables are codified by some simple membership functions as shown in Figure 4 and output membership functions are shaped with singletons. It is worthy to mention that the membership functions were selected intuitively at the beginning. Nevertheless after some preliminary experiments during the game development, they were tuned for a smoother game play.

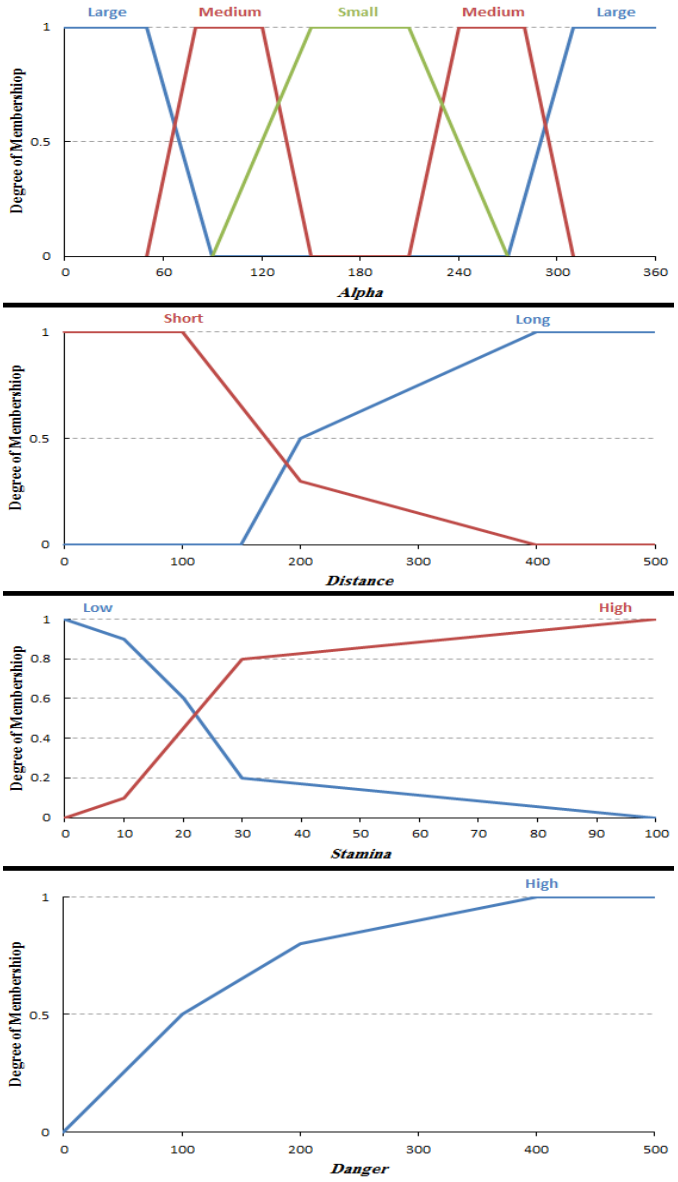

Figure 4: Fuzzy Membership Functions

Near Fuzzy System rules are shown in Table 3, where columns represent the variable *OppStamina* and rows represent the variable *MyStamina*.

Table 3: *Near Fuzzy System* Rules

OppStamia / *MyStamina*	***Low***	***High***
Low	*W_MyHandDist* = 0.3**Attack*/5 **AND** *W_OppHandDist* = 0.3**Flee*/5 **AND** *W_Distance* = -0.3**Flee*/*Attack* **AND** *W_MyStamina* = 1	*W_MyHandDist* = 0.3*5/*Flee* **AND** *W_OppHandDist* = 1 **AND** *W_Distance* = -0.5**Flee*/5 **AND** *W_MyStamina* = 1
High	*W_MyHandDist* = 1 **AND** *W_OppHandDist* = 0.3**Attack*/5 **AND** *W_Distance* = 0.5**Attack*/5 **AND** *W_MyStamina* = 0	*W_MyHandDist* = *Attack*/5 **AND** *W_OppHandDist* = *Flee*/5 **AND** *W_Distance* = 0.3 **AND** *W_MyStamina* = 0.3

As an example in the case where both *OppStamina* and *MyStamina* are *low* the following rule will be triggered.

IF (*MyStamina* = *Low* **AND** *OppStamina* = *Low*) **THEN** (*W_MyHandDist* = 0.3**Attack*/5 **AND** *W_OppHandDist* = 0.3**Flee*/5 **AND** *W_Distance* = -0.3**Flee*/*Attack* **AND** *W_MyStamina* = 1)

Far Fuzzy System rules are shown in Table 4, where columns represent the variable *Distance* and rows represent the variable *Alpha*.

Table 4: *Far Fuzzy System* Rules

Distance / *Alpha*	***Short***	***Long***
Low	*W_Distance* = 0.1 **AND** *W_Alpha* = 0 **AND** *W_Danger* = 1	*W_Distance* = 1 **AND** *W_Alpha* = 0 **AND** *W_Danger* = 1
Medium	*W_Distance* = 0 **AND** *W_Alpha* = 1 **AND** *W_Danger* = 0.7	
Large	*W_Distance* = 0 **AND** *W_Alpha* = 1 **AND** *W_Danger* = 0	

Finally, *Overall Fuzzy System* rules are as follows:

- **IF** (*Distance* = *Short*) **THEN** (*W_NearFuzzySystem* = 1 **AND** *W_FarFuzzySystem* = 0)
- **IF** (*Distance* = *Long*) **THEN** (*W_NearFuzzySystem* = 0 **AND** W_FarFuzzySystem = 1)
- **IF** (*Danger* = *High*) **THEN** (*W_NearFuzzySystem* = 0 **AND** W_FarFuzzySystem = 0.8)
- **IF** (*Alpha* = *Large*) **THEN** (*W_NearFuzzySystem* = 0 **AND** *W_FarFuzzySystem* = 0.7)

W_parameter denotes the weight and importance of the *parameter*, where the *parameter* can be one of the variables *MyHandDist*, *OppHandDist*, *Distance*, *MyStamina*, *Alpha* and *Danger*; A negative value of *W_Distance* states to increase the distance and a positive value states to decrease it. In addition, *W_FarFuzzySystem* and *W_NearFuzzySystem* are the degrees of truth of the statements, inferred from the outputs of *Far Fuzzy System* and *Near Fuzzy System* respectively. This is how the *Overall Fuzzy System* affects the whole decision making process. *Flee* and *Attack,* which can be configured by user between 1 and 10, specify the preference of the boxer for escaping or attacking. These two parameters can be set to 5 for a typical boxer with normal behavior. Some important notations are summarized in Table 5.

Table 5: Nomenclature

Notation	Short Definition
Alpha	Degree between two boxers
Attack	Tendency of AI boxer to attack
Danger	Danger of being near of the ring border
Distance	Distance between two boxers
Flee	Tendency of AI boxer to escape
MyHandDist	Distance of AI boxer's hand to its opponent
MyStamina	AI boxer's strength to continue fighting
OppHandDist	Distance of the opponent's hand to AI boxer
OppStamina	Opponent's strength to continue fighting

Finally, in the actuation layer, if the previous layer demands a punch, it will throw a punch based on the abilities of its

hands. Otherwise it should test all possible actions on its own ring simulator during two clocks. Then a fitness factor is calculated for each action based on the weighted parameters, which have been produced by the Fuzzy system. In fact, the differences in the values of the *parameters* (such as $\Delta Alpha$) during simulation are multiplied by their corresponding *W_parameter* (like *W_Alpha*) and divided by a normalizer coefficient. The sum of these resultant values determines the fitness of each action. Additionally, in the actuation layer, there are some tricks to avoid getting stuck somewhere in the ring, or persisting in just punching in a row. For instance, if the boxer retains its position near the ring edges for a while, the AI ignores the current situation of the game and moves on to increase its distance from the sides of the ring.

RESULTS

The proposed AI system has been applied to the introduced game engine. The performances of five Fuzzy systems with different configurations were assessed against a scripted AI and 10 simulations have been performed over each one. Since there is no standard method for gaging the 'believability' of game bots (Gorman et al. 2006), only the match results (wins and losses) are illustrated in Table 6.

Table 6: Game Results of the Fuzzy AI against the Scripted AI

Fuzzy AI Type	Win (Knockout)	Win (Timeout)	Lost (Knockout)	Lost (Timeout)
Balanced (*Flee*=*Attack*=5)	60%	10%	10%	20%
Defensive (*Flee* = 10, *Attack* = 1)	0%	10%	60%	30%
Offensive (*Flee* = 1, *Attack* = 10)	80%	0%	20%	0%
Fairly Defensive (*Flee* = 7, *Attack* = 3)	60%	10%	20%	10%
Fairly Offensive (*Flee* = 3, *Attack* = 7)	70%	20%	10%	0%

Based on the empirical perception about the AI agents, the most significant difference between the playing methods of the Fuzzy AI and the scripted AI is the diverse behavior of the Fuzzy AI. This matter becomes more obvious when two scripted AIs play against each other. In this case, it is seen that many repetitive situations happen in each run of the game. But when one player utilizes the Fuzzy AI, the game gets more unpredictable; In fact, the Fuzzy agent insists less on a specific action and has better adaptation to the game.

CONCLUSIONS

By using a Fuzzy system, rules and membership functions designed by a human expert, we can avoid cheating and just follow the natural laws of the game. This could be very useful for extending and making better games, i.e. when a human finds a new rule which can be useful in a game, we will be able to easily insert the new rule into the game as a Fuzzy rule.

The main benefit of the Fuzzy logic approach to game development is human-like behaviors, which guarantee believable and natural behavior (not necessarily perfect) and increase the satisfaction of human players. Another advantage of using Fuzzy logic is that, unlike scripted AI, a developer does not need to consider all possible situations in a game. Furthermore, the idea of various Fuzzy subsystems divides the state space of the problem, which makes the scaling to more demanding settings easy and could be efficiently exploited in other genres of games. As an example, in RTS games as the most AI challenging type of games, two simultaneous main concerns are consuming the resources for building different structures and training the army forces for attacking the opponent. To fulfill this purpose, two Fuzzy systems can be used for handling these two problems. In addition, a third Fuzzy system could be used for determining the priority of each of those Fuzzy systems, according to the current situation of the game.

Future enhancements are required to overcome the deliberate delay, which is applied to the AI; a simple solution would be using the probabilistic Bayesian methods in order to predict the current situation of the opponent. Another possible outcome of the *Clever Boxer* is a boxing trainer, just like a human boxing coach; The AI system watches a boxing game and then suggests how the player could improve its performance based on its own Fuzzy human-like rules.

REFERENCES

Bourg D. M. and Seeman G. 2004. *AI for Game Developers*. O'Reilly, chap. 8.

DeLooze L. L. and Viner W. R. 2009. "Fuzzy Q-Learning in a Nondeterministic Environment: Developing an Intelligent Ms. Pac-Man Agent". In *IEEE Symposium on Computational Intelligence and Games (CIG'09)*, pp. 162-169.

Doss P. 2006. "Artificial Intelligence in Computer Games". Bowie State University, M. Sci. Thesis.

Johnson D. and Wiles. J. 2001. "Computer Games with Intelligence". In *Proceedings of the 10th IEEE Conference on Fuzzy Systems* (Dec. 2-5), vol. 3, pp. 1355- 1358.

Gorman B. et al. 2006. "Believability Testing and Bayesian Imitation in Interactive Computer Games". In *Proceedings of the 9th Int. Conf. on the Simulation of Adaptive Behavior (SAB'06),* Springer, vol. LNAI.

Li Y. et al. 2004. "Fuzzy Logic in Agent-Based Game Design". In *Proceedings of the Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS '04)*, pp. 734-739.

Loiacono D. et al. 2010. "The 2009 Simulated Car Racing Championship". In *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, no. 2, pp. 131-147.

Onieva E. et al. 2010. "Overtaking Opponents with Blocking Strategies Using Fuzzy Logic". In *IEEE Conference on Computational Intelligence and Games (CIG'10)*, pp123-130.

Perez D. et al. 2009. "Evolving a Fuzzy Controller for a Car Racing Competition". In *IEEE Symposium on Computational Intelligence and Games (CIG'09)*, pp. 263-270.

Rollings A. and Ernest A. 2006. *Fundamentals of Game Design.* Prentice Hall, chap. 13.

Russell S. and Norvig P. 1995. *Artificial Intelligence: A Modern Approach*. Prentice Hall.

Takagi T. and Sugeno M. 1985. "Fuzzy identification of systems and its application to modeling and control". In *IEEE transaction on systems, man and cybernetics,* vol. 15, no. 1, pp. 116-132.

Tozour P. 2002. *The Perils of AI Scripting*. In Rabin S. *AI Game Programming Wisdom,* Charles River Media, pp. 541-547.

Zadeh L. 1965. "Fuzzy sets". *Inf. Control*, vol. 8, pp. 338-353.