Mimicking Human Strategies in Fighting Games using a Data Driven Finite State Machine

S. Saini, P.W.H. Chung and C. W. Dawson
Department of Computer Science,
Loughborough University,
Loughborough, UK, LE11 3TU
S.S.Saini@lboro.ac.uk

***Abstract*—Multiplayer fighting videogames have become an increasingly popular over the last few years, especially with the introduction of online play, making for a more competitive experience. Multiplayer fighting games give players the opportunity to utilize particular strategies and tactics to win, allowing them to use their own signature style. As a player can only play against a particular opponent who is actively participating in the game themselves, they cannot practice combating the opponent's style if the opponent is not participating in the game. This paper presents a novel approach for an avatar to learn and mimic the style of a player. It does this by recording and analyzing the data before splitting it up into two tiers; tactical data and strategic data.. The approach uses a Naïve Bayes classifier to classify the tactics to particular states, and a Data Driven Finite State Machine to dictate when certain tactics are used. Statistics recorded during an experiment involving the approach are discussed, which indicate that the architecture of the Artificial Intelligence is fit for purpose, but does require refinement. Limitations of the architecture are discussed, including that such an approach may not provide accurate results when more parameters are considered.**

Keywords-Artificial; Intelligence; fighting; game; FSM; Bayes

I. INTRODUCTION

Gaming has seen a large increase in popularity over recent years, owing largely to the availability of online gaming. Increased usage of multiplayer functionality has brought new challenges to research Artificial Intelligence (AI) within gaming. This field of research is commonly referred to as Game AI . Fighting games have been explored in the past, however, the majority of research is restricted to creating a 'good' AI player. Reference [2] investigate the use of Artificial Neural Networks to create an AI fighter. However, as with much of the research conducted in the field of Game AI, the problem is concerned with improving the AI player such that it is harder to beat, rather than refining the AI to behave in a particular way defined by a human.

There is a lack of research conducted in the field of AI applied to strategic fighting games. While the use of AI techniques make for engaging Real Time Strategy games [1], the work carried out in the genre of fighting games is limited to shorter term tactics using ANN [2]. Implementing AI techniques in a fighting game to enable the CPU controlled player to learn and mimic human strategies is an area that has not yet been explored.

The research reported in this paper is concerned with implementing an AI Player that is capable of mimicking human tactics and strategies. Strategy can be defined as a preliminary decision making activity, whereas tactics can be defined as an action based decision making activity [7]. This is to say that a strategy is a long term plan formulated ahead of time, where as tactics are short term actions that are carried out amidst the action taking place, to realize the strategy. In the context of fighting games, this would firstly entail executing the same combinations of moves as the human subject being mimicked, which addresses the tactical level. Secondly, this would need to happen in response to a situation where the moves are carried out based on the statistics of the game world, which addresses the strategic level.

The following sections of this paper shall describe the proof of concept game used to aide this research, and also provide some background on AI techniques used in the solution presented. An overview of the solution as well as the game that has been designed as a test bed is also provided.

II. PROOF OF CONCEPT GAME

To aid the design, implementation, testing and evaluation of a sufficient approach to address the problem, a proof of concept game has been created. The game is a one-on-one fighting game, allowing players to perform attacks, movements and defend. In the game, each fighter has a health attribute initiated at 100. If a fighter's health attribute reaches zero, the other fighter wins the bout.

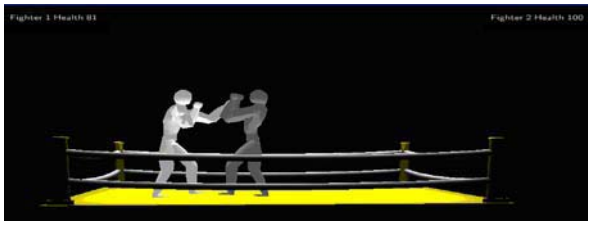

Figure 1. Proof of Concept Game screenshot.

Table I below lists the moves available to a player as well as the effect a move has on the opponent, provided the distance between the two on-screen fighters falls within the 'From' and 'To' threshold.

978-1-4244-8625-0/11/$26.00 ©2011 IEEE

TABLE I. GAME RULES

Move	From	To	Health	Blocked	Evasion	Notes
Jab	4.1	5	1	Health – 0.5	Back	
Cross	4.1	5.5	2	Health – 0.5	Left	
Right Hook	4	4.7	3	Health – 0.5	Back	
Left Hook	4	4.7	3	Health – 0.5	Back	
Uppercut	0	4	4	Health – 0.5	Right	
Haymaker	4	4.5	10	N/A	N/A	
Right Body Shot	0	4	2	Health – 0.5	N/A	
Left Body Shot	0	4	2	Health – 0.5	N/A	
Short Jab	0	4	2	Health – 0.5	Back	
Short Cross	0	4	3	Health – 0.5	Left	
Evade Back						Evasion
Evade Left						Evasion
Evade Right						Evasion
Push	0	4	2	Health – 0.5		Opponent 5 back
Block						
Low Block						
Front Kick	4	4.7	2	Health – 0.5		
Low Kick	0	4	2	Health – 0.5		
Sidekick	4.1	5.5	4	Health – 0.5		
Roundhouse	4.1	5.5	2	Health – 0.5		
Stomp Kick	4.1	5.5	4	Health – 0.5		
Knee	0	4	2	Health – 0.5		
F Lunge						Player 6 Forward
B Lunge						Player 6 Back

If the opponent is within the range specified by the 'to' and 'from' attributes listed, and is not blocking or performing an appropriate evasion, they shall be struck and the value in the 'health' field shall be deducted from their health. If the opponent is performing a block (or in some cases a low block) when the move connects, their health shall deplete as indicated by the value 'blocked' field. If timed correctly, certain moves can be evaded. For example, if the player throws a jab and the opponent reacts by performing the 'back' move with the correct timing, the move shall not connect and no health shall be depleted.

The rationale behind designing a game in this way was to allow players to combine their own unique tactics to form longer term strategies. The variety of moves include lunging forward and back, making for flexibility in movement. This footwork combined with the evasion maneuvers and attacks make for a creative fighting system, empowering the players to define various strategies and providing them with the tools to execute short term tactics to accommodate said strategies.

III. BACKGROUND

This section provides background knowledge on the AI techniques used as part of the multi-tiered architecture.

A. *Naïve Bayes Level*

Various classifiers can be used for clustering traits to certain player types. In the same manner, classifiers can be used to cluster moves or combinations of moves and assign them to tactics or strategies. The Naïve Bayes Classifier (NBC) is one such classified, and provides a simple approach to classification which simplifies the problem by assuming attributes are independent of the target value. The problem typically involves a set of training data, then a new instance the classifier is asked to produce a target value for using (1).

$$v_{NB} = \underset{v_j \in V}{\arg\max} P(v_j) \prod_i P(a_i \mid v_j) \quad (1)$$

where v_{NB} is the class value output by the classifier, and a_i are the values for attributes fed into the classifier. v_j denotes elements of the set V which are the possible class values. For example, in the context of the proof of concept game,V = {Inner, Outer, Defend}. The NBC is typically less accurate than Bayesian Belief Network due to its ignorance, however, it is computationally quicker [3].

B. *Data Driven Finite State Machine*

In practice, a Finite State Machine is a description of how an object can change its state over time in response to the environment and events that occur. Each state in the FSM represents a behavior, resulting in AI behavior changing as states change from one to another. The function T resides across all states, meaning that the states shall be left and entered in accordance to fulfilling the transition criteria for that particular state. The input is fed into the FSM continually as long as the game is active [4].

The use of finite state machines in videogames is promoted by many developers due to their robust nature as they are easy to test and modify [5]. However, the primary limitation of finite state machines lies in its predictability. The actions performed in a given state do not alter as time goes on, nor do the triggers that cause state transitions. This is to say that the entire finite state machine is a static, rule based system [5], rather than a system that is capable of learning and evolving as the game is played. Once a player has found a way to counter the finite state machine logic, they could exploit the static nature of the technique and use the same tactics to succeed each time. One may argue that finite state machines are not representative of a valid artificial intelligence technique as they do not adapt or learn from their environment.

The static and predictable nature of hard-coded finite state machines can be addressed by implementing data driven finite state machines. The data driven approach uses authored data that powers the FSM. A data driven FSM is useful for instantiating custom FSMs whose states and transition logic are defined in an external file [6]. This approach of placing a dependency on an external file to dictate how the FSM should behave makes for a flexible solution. If we consider simulating a players' strategy, the data contained within the file can be written in real-time during gameplay and then used to compile a finite state machine.

IV. IMPLEMENTATION

A. *Solution Architecture*

The approach used to solve the problem relies on a combination of the techniques discussed previously.

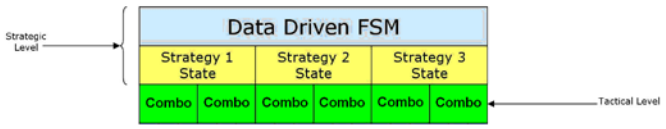

Figure 2. Multi-tiered architecture

By identifying the levels of play into strategic and tactical, a specific AI technique can be used to tackle each level, with information being passed between levels. Figure 2 shows the architecture for this approach. A data driven finite state machine (FSM) is used to model the players' various strategies and how/when the player transitions into a particular strategy. While the FSM was previously cited as being a weak technique due to predictability, and lack of flexibility at the tactical level, a data driven FSM rectifies these weaknesses.

The usage of the architecture can be categorized into two distinct approaches. The first use would be during the data capture phase. This is when the human vs human bout takes place. It is during the data capture phase that information on the moves performed as well as the condition of the game world (namely the player's health and distance between fighters) is collated. Once this information has been identified, the moves that are performed (see Table 1 for list of available moves) are assigned to different pre-determined states using the Naïve Bayes classifier that has been trained to classify such data. In the proof of concept game, there are three states; Outer, Inner and Defend. The Outer state is for moves and combinations of moves that are executed at a distance, whereas the inner state is for moves and combination of moves that are performed up close to the opponent. The defend state is used for combinations that are deemed defensive and entail heavy blocking.

Once the combinations have been assigned to their respective strategy states, the Finite State Machine is created based on the inputs captured during the human vs human bout. The tactics are classified and state transitions are identified along with transition functions. The data driven finite state machine contains the previous, current and next state for a given transition and is based on the health of the AI player.

During the simulation phase of the architecture usage, information is passed from the top down. Strategies are selected based on inputs from the DD FSM. These strategies dictate the tactics that are used as they have previously been classified during the data capture phase. Once a tactic is actioned, the appropriate moves are carried out by the AI player.

B. *Experiment and Results*

To demonstrate the effectiveness of the proposed approach, a strategy is fabricated prior to playing the game. The strategy being actioned here is as follows:

- The fighter shall begin by maintaining a distance and attacking the opponent using long range moves (During this time the fighter is in the Outer state).
- If the fighter is being pummeled to the point that their health statistic drops beyond a certain point (circa 70), they shall retreat and assume a defensive position (during this time the fighter is in the Defend state).
- Whilst blocking, the fighter shall lose further health. When the health depletes beyond a certain threshold, which in this case is approximately 50 units, the fighter shall attack at close quarters (During this time the fighter is in the Inner state).

Two human players play the game, with the second player employing the aforementioned strategy. The raw data from the bout is recorded, including the tactics (combinations of moves) used during each phase of the strategy. These data are transformed through a series of programs. Firstly, the tactical combinations are classified to strategy states, outer, inner or defend, using a Naïve Bayes classifier. Following the classification, the data driven finite state machine is created. Each state in the FSM has the appropriate tactics assigned

based on the raw data collated during the human vs human bout. The data in Table III and Table IV were extracted from raw data that were captured in real-time during gameplay. The statistics of the game as well as the moves being carried out were spooled to a file every time the AI fighter made a move, or in the instance of the human vs. human bout, whenever the player being mimicked made a move (refer to Table II for legend on moves).

TABLE II. MOVES LEGEND

Character	Move
j	Jab
c	Cross
b	Block
m	Left Body Blow
n	Right Body Blow
u	Uppercut
a	Back Lunge
z	Back Evasion

TABLE III. HUMAN VS HUMAN STATISTICS

Health	Moves	State
100	j j	Outer
100	j	Outer
100	c c	Outer
100	j j c	Outer
67	b b b	Defend
50	b	Defend
50	b	Defend
49.5	m	Inner
49.5	n	Inner
49.5	u	Inner
49.5	u u	Inner
49.5	m m	Inner

TABLE IV. AI VS HUMAN STATISTICS

Health	Moves	State
100	j j	Outer
89	c	Outer
89	j j	Outer
79	j	Outer
70	c c	Outer
67	b b	Defend
57	b b	Defend
56	b	Defend
55.5	b b	Defend
45.5	b	Inner
45.5	n	Inner
17.5	m m	Inner
17.5	u	Inner
17.5	u	Inner

Table III above shows the overall trend of the second fighter, whose strategy shall be mimicked. The data show that the second fighter begins the bout whilst delivering long range attacks. When the fighter's health drops below 67, they begin blocking, which in turn depletes their health at a slower rate. When the health is depleted beyond 50, the second fighter begins attacking again, this time moving in close and using shorter range attacks. This strategy and the underlying tactics are mirrored in the bout between a human player and a CPU controlled player which is based on fighter 2 from the human vs human bout. As Table IV shows, the same tactics are used for each of the states, and the transition to different states occur at around the same threshold values.

In Table V and Table VI, different strategies are used and data are recorded at 10 point health intervals. The Human Moves and Human State columns contain moves carried out by the human being mimicked at the instant the health dropped 10 points, as well as the state they correspond to. These data are recorded during the initial human vs human bout. The AI Moves and AI State columns contain moves carried out by the AI during the simulation at the instant the health dropped 10 points, as well as the state they correspond to.

In Table V, the strategy employed by the human player who is to be mimicked is as follows :

- Initially perform close range moves from the inner state.
- When health drops below 65, begin attacking from a distance (outer state).
- If health drops below 35, start blocking (defend state).
- When health drop below 15, revert to inner state.

In Table VI, the strategy employed by the human player who is to be mimicked is as follows :

- Begin in the defensive state, perform blocks, back lunges and evasions.
- When health drops below 50, begin attacking from the outer state.

TABLE V. COMPARISON 1

Health	Human Moves	Human State	AI Moves	AI State
100	u	Inner	n m	Inner
90	n n	Inner	n	Inner
80	m m	Inner	n	Inner
70	u m	Inner	m m	Inner
60	j j	Outer	c	Outer
50	c	Outer	j j	Outer
40	j	Outer	j	Outer
30	b	Defend	b b	Defend
20	b b	Defend	b b	Defend
10	n	Inner	n	Inner

TABLE VI. CAMPARISON 2

Health	Human Moves	Human State	AI Moves	AI State
100	z	Defend	b	Defend
90	b	Defend	b b	Defend
80	a	Defend	z	Defend
70	z	Defend	b	Defend
60	b	Defend	a	Defend
50	b	Defend	b b	Defend
40	c	Outer	j	Outer
30	c c	Outer	c c	Outer
20	j c	Outer	c j	Outer
10	j j	Outer	j	Outer

The data provided in Table V and Table VI demonstrate that the state transitions made by the AI player are driven by the same transition functions as those made by the human. The tactics used by the AI within a particular state do not deviate from those used by the human when in that state throughout the course of the game.

V. CONCLUSIONS AND DISCUSSION

The data provided in Table III and Table IV correspond with the strategy that was premeditated from the outset. This is also the case for the strategies captured in Table V and Table VI. These data demonstrate that the tiered approach described in this paper can indeed be utilized to mimic human strategies. The results show the tactics that are performed by the CPU fighter correlate to those performed by the human fighter.

While the approach in this paper has been demonstrated to work, it is not without its limitations. The primary limitation of the technique described in this paper lies with the fact that it cannot be tailored to games that utilize a variety of statistics. Some fighting games may consider the player's morale and stamina as factors on which strategic decisions are based. However, due to the nature of the data driven finite state machine, if a variety of statistics change at the time of transitioning into a different state, there is no way of knowing why a player chose to enter the state as the decision could be based on any of those statistics dropping below a particular threshold, or even all of them. Further to this, the strategy takes into account the actions performed by the fighter that is to be mimicked. There is an underlying assumption that the driver for these actions is solely the fighter's health. There is no consideration for what the opponents last move was, or what state the opponent is in. It could very well be the case that the player in question is waiting for their opponent to begin blocking high before attacking their lower body. However, this strategy would not be captured using the current model as the trigger to entering the attack state would be dependent on the opponent's actions rather than any statistics.

A further limitation of the technique discussed in this paper involves anomalies that may exist in the data. The strategy described and mimicked here was carefully thought out and executed during gameplay. However, if a player has a strategy in mind and deviates from it, whether by mistake or intentionally, the consequences of adding such noise to the data could impact the overall strategy of the AI player. Noise reduction could be used during the transformation of data to detect and handle such anomalies so that they do not impact the high level strategy being used.

Furthermore, playing the game against an AI fighter can feel static and not as fluid as it does when playing against the human fighter. This is because the strategy is being mimicked exactly, with no consideration for mistakes and preferences. For example, if a player has a combination they enjoy executing time and time again then this is not evident in the bout against the AI Player due to the indiscriminate way that the tactics belonging to a state are chosen. Furthermore, the player may occasionally make mistakes and perform certain moves when it was not their intention. Rather than treating these data as an anomaly and disregarding them, they can be used to create a concise Player Model, adding a level of realism and chaos to the way the AI fighter plays the game. A Player Model is a model based on statistic of how a particular player plays a game and considers their strengths, weaknesses, preferences and actions they may avoid. Further research could be conducted to append a player model to the architecture of the solution described in this paper. A similar experiment could then be conducted, but further to the statistical analysis conducted here, a Turing type test could also be conducted.

REFERENCES

[1] C. Miles, J. Quiroz, R. Leigh and S. J. Lewis, "Co-evolving influence map tree based strategy game players," in Proc. 2007 IEEE Symposium on Computational Intelligence in Games, pp. 88-95, 2007.

[2] B. Cho, S. H. Jung, Y. R. Seong and H. R. Oh, "Exploiting intelligence in fighting action games using neural networks," IEICE Trans. on Information and Systems, vol. E89-D, no. 3, pp. 1249–1256, 2006.

[3] S. He, J. Du, J. Meng, H. Chen and Q. Zhu, "Strategy-based player modelling during interactive entertainment sessions by using Bayesian classification," in Proc. 4th International Conference on Natural Computation, pp. 255-261, 2008.

[4] D. Fu and R. Houlette, "The Ultimate Guide to FSMs in Games," in A.I Game Programming Wisdom 2, S. Rabin, Ed. Charles River Media, 2003, pp. 283-301.

[5] D. Johnson and J. Wiles, "Computer Games with Intelligence," Australian Journal of Intelligent Information Processing Systems, vol. 7, pp. 61-68, 2001.

[6] G. Rosado, "Implementing a Data-Driven Finite State Machine," in A.I Game Programming Wisdom 2, S. Rabin, Ed. Charles River Media, 2003, pp. 307-317.

[7] A. Mouchet. "Subjectivity in the articulation beetween strategy and tactics in team sports: an example in rugby," Italalian Journal of Sport Sciences, vol. 12, no. 1, pp. 24-33, 2005.