A Fuzzy Logic Based Attack Strategy Design for Enemy Drones in Meteor Escape Game

Ali Emre Soylucicek, Erkan Bostanci, and Aykut Burak Safak

***Abstract*—This paper presents the use of fuzzy logic in designing the enemy behaviour for a popular arcade game called Meteor Escape. The original game was played against a cloud of meteors that might collide with a spaceship which is trying to make its way in the deep space. The game is extended with additional Kamikaze drones working with a simple decision mechanism based on zero-order Sugeno model. The results show that this additional feature improves the gameplay.**

***Index Terms*—Fuzzy logic, arcade game, enemy design, Unity 3D.**

I. Introduction

Artificial intelligence is a fundamental method for creating opponents in games [1]. The user might play the game alone and the game has to provide a significant level of challenge. This can be provided by creating enemy units with artificial intelligence, allowing the game to be played against the computer.

In the past days, creating AI in games were harder. Old technology and low processor speeds were creating big problems for game developers. All the graphics and game engine had to be processed by the same processor, developers didn't have enough process power for creating good AI systems. But thanks to the current technology, game graphics are processed by separate graphics processors (GPUs) providing an enormous power for AI [2].

It is known that the purpose of game AI is to make the game more challenging, but not unbeatable, hence more interesting. The enemies in a game can be programmed to know the user position in advance, before line of sight, making it impossible to beat the enemies. AI, here, should be designed so that it doesn't overwhelm the player while adding challenging parameters to the game. This feature of AI actually what makes a game success [3].

There are many different ways of creating enemies with AI. State machines can be used for creating AI. State machines includes specific states for related occasions. In the game loop these states can change according to the gameplay or according to the rules. The drawback of this method is that it can be difficult to improve and change the system. Certainly, state machines are widely used inside most popular games, but it is not the only way to do it. Fuzzy logic can also be implemented within the game AI [4] which can provide more human-like decisive behaviour.

Manuscript received August 18, 2016; revised December 8, 2016.

The authors are with SAAT Lab in Computer Engineering Department, Ankara University, Turkey (e-mail: alisoylucicek@gmail.com, ebostanci@ankara.edu.tr, a.buraksafak@gmail.com).

Fuzzy logic is not widely used inside games. Some implementations are made in some AI systems inside games, but they are not very popular. But there are some examples which most of the gamers know.

Unreal is a very popular first person shooter game. Enemy AI inside the game was implemented with fuzzy logic [5]. This made the enemies in game more intelligent and less predictable. Players really enjoyed the enemies inside the game. S.W.A.T. 2 is a real time strategy game [6] in which the characters that are controlled by the game (Non-Player Character, NPC) using fuzzy logic based AI. Enemy Nations is another example of fuzzy logic implemented AI. Rules and the inference engine is separated from each other for better enemy behaviour [7].

As it can be seen from these examples, fuzzy logic can work inside AI systems and they provide very good results. This paper presents the fuzzy-logic based enemies for a popular game. The rest of the paper is structured as follows: Section II gives general information about fuzzy logic and its principles, followed by Section III cases where possible caveats of such systems can arise are discussed along with their advantages. Section IV gives a brief information about Unity 3D which was used for implementing the game. Section V describes the fuzzy logic system used in the game and the results for this is presented in Section VI. Finally, the paper is concluded in Section VII.

II. Fuzzy Logic

Fuzzy logic was presented by Lotfi A. Zadeh in 1965. Conventional logic in computers have two consequents: true or false, facilitating a bimodal structure for decision making. However, this method is far from the way used in human thinking. In real life, people may have levels to decide an occasion by constructing statements like 'it is *really* hot today. 'I *usually* read books'. From these examples it can be seen that humans think in a fuzzy way, *i.e.* we use words of uncertainty to describe behaviours or events [8], [9].

A fuzzy decision making engine comprises of 3 main steps: Fuzzification, inference and defuzzification. Fuzzification step involves obtaining inputs and converting them to linguistic variables. For example: a person who is 180 cm height can be considered as a tall person. So this person is a member of the group of tall people. In fuzzy logic, this group is called a *fuzzy set*. Fuzzy sets describe a degree of a variable in linguistic way. From previous example, a 180 cm person is member of "Tall" fuzzy set.

After fuzzifying crisp inputs to linguistic variables, a type of inference must be chosen. In this case, we will focus on Sugeno type fuzzy inference. Sugeno type fuzzy inference is developed by Takagi, Sugeno and Kang [6]. Sugeno type inference uses a single spike, *a.k.a.* singleton. This singleton is the consequent of the related fuzzy rule.

The inference engine must check the rule base, which is the database of rules of the behaviours. These rules are prepared for the behaviours of enemies inside the AI system. The rules in this system have two antecedents and a single consequent. Antecedents are the linguistic variables and their membership values. If the antecedents of the specific rules are right, then that specific rules are fired. Structure of a rule is as follows:

- *if x is "A" OR y is "B" then z is f(x,y)*

Fig. 1 depicts the rule evaluation process. Two antecedents are calculated in "OR" operator, which takes maximum of two membership values. The taken value is multiplied by the constant *k1* which can be seen as a consequence of the decision making process. All of the fired rules create an output value, and average of these values is the output value. This output value can be used according to the needs of the AI system.

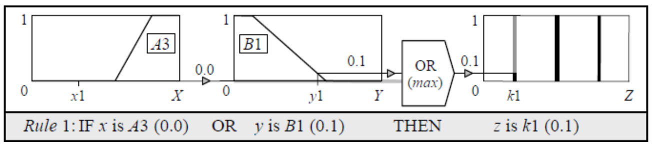
1
A3
0
x1
X
0.0
1
B1
0
y1
Y
0.1
OR
(max)
0.1
1
0
k1
Z
Rule 1: IF x is A3 (0.0) OR y is B1 (0.1) THEN z is k1 (0.1)

Fig. 1. Sugeno Type Rule Evaluation [2].

Fuzzy logic systems are very useful in behaviour-based systems and decision making. For example, enemy can guess how scared the opponent is from its movements, or how strong the opponent is and hence it can decide next action accordingly. Also there are more implementations of fuzzy logic and neural networks for more natural and realistic behaviour AI enemies inside games.

III. Pros and Cons of Fuzzy Decision Making

Implementation of fuzzy logic inside AI system is quite different from other type of logics. Most of the AI systems are created by state machines. Creating states according to the behaviours of enemies inside a game. But the number of states might be overwhelming in some implementations. And also maintenance of state machines is really hard and time consuming.

But in fuzzy logic, rules are separated from the inference engine and all the rules can be changed much easier. Easy maintenance means less time consumed for development of the system, which is more efficient. Also these rules can be decided without a programmer. Rules can be decided in a linguistic way, then a programmer can implement it inside the rule base. Yet, these rules must be prepared with caution. Someone with game design experience should prepare the rules. Otherwise, rules can conclude incrorrect solutions, which makes development progress longer.

A drawback of fuzzy logic system is the dependence of inputs. If there are a lot of inputs to the inference engine, rules are increased exponentially. This makes calculations harder and longer. The fuzzy system needs to be created according to the inputs, or should be separated to different sections and work sequentially.

IV. Unity 3D

Unity 3D is a cross-platform free game engine. It features a lot of components which makes creating games easier. Having its own physics engine and image implementations makes it a powerful tool. Unity supports C#, Javascript and also Lua as the programming and scripting languages.

Engine features are not the only thing which makes Unity 3D a powerful tool. Also the trainings provided by Unity's website [10]. There are lessons about scripting, animations and even physics. Hours of lessons and documentation makes this game engine one of the best tool for game developers.

In addition to this, the community of Unity is huge. There is a dedicated website for the community questions and answers. Developers can ask questions about their problems and more experienced developers offer their solutions and this is really good for beginner developers.

Unity has a clean and organized user interface. There is a scene view to see all the objects in the game scene. Also there is a game pre-view to see how users will be seeing the game. Game view only sees what the main camera is looking at the scene. It is almost similar to making a movie which you can participate. Hierarchy panel is the objects in the scene and all the objects are well organized in child and parent hierarchy.

Every object in the game engine is a Unity Game Object, which can have multiple components. From physics components to renderer components, there are a lot of choices for the programmer's needs.

The game in this paper is developed inside Unity3D engine and fuzzy logic is implemented inside enemy AI.

V. Fuzzy Enemy Design

This is a space themed 2D arcade game to implement fuzzy logic in enemy AI. Game components are user spaceship, enemy drones and meteors as depicted in Fig. 2. Objective of the game is to stay alive as long as possible.

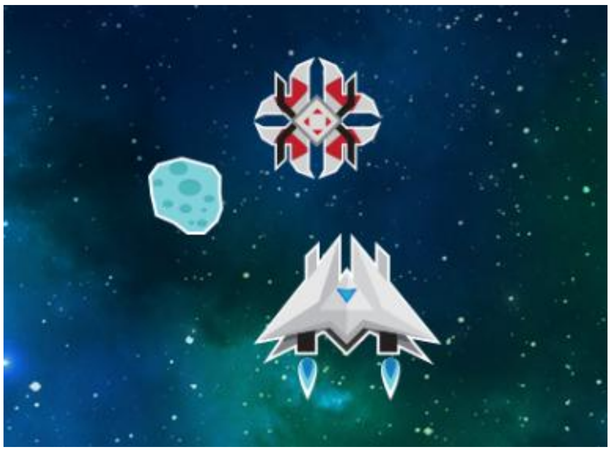

Fig. 2. Game objects

All of the designs of game objects were prepared in Adobe Illustrator program. Background photo is edited in Adobe Photoshop. Designs were created as simple as possible, because main objective is to show how fuzzy logic works inside enemy AI.

The spaceship is the game object which user controls. The movement of the spaceship is prepared due to the movement in free space. Thrusters can only move ship forward, to turn and move the ship to another direction, user must use thrusters and the rotation. User spaceship also has 100 life points before it is destroyed.

Meteors have low level artificial intelligence. User spaceship has a gravitational pulling force at certain radius. When a meteor enters this radius, it follows user spaceship. When it gets out of the pull radius, it continues to move like it would do in empty space.

Enemy 'Kamikaze' drones are the most complicated objects in the game. Fuzzy logic was implemented inside these drones. These drones have a radar of certain distance radius. This radar has an operation frequency as will be described later. When the user spaceship enters this radar, fuzzy logic inside the drone calculates the position and distance between itself and the user spaceship. After fuzzy logic calculations, it shoots itself into a direction and explodes as in Fig. 3. The main purpose is to destroy the user spaceship.

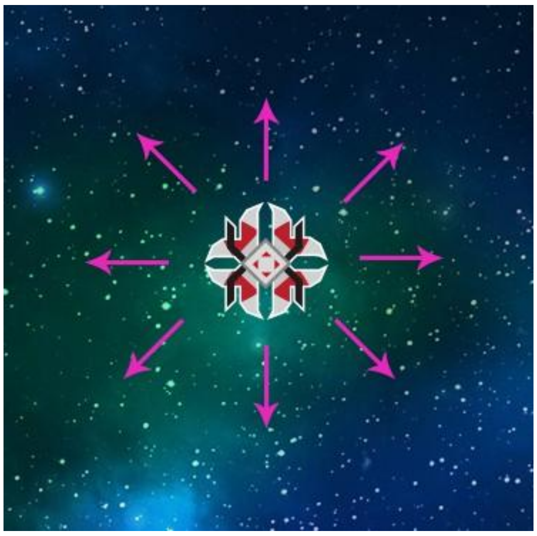
Fig. 3. Kamikaze drone motion directions.

For the explosion of enemy drone, a free 2D explosion asset package was used from Unity asset store. Explosions have certain radius of effect, and if user spaceship is inside this radius, ship loses 20 life points. This free asset package already has the particle effect and the animations ready. In the code of the enemy drone controller, when it reaches maximum range or gets close enough to the user spaceship, this explosion is spawned at the location of enemy drone. When explosion occurs, an explosion force is applied to the user spaceship to give a more realistic effect.

For the implementation of fuzzy logic inside enemy drones, input and output membership functions are created. Membership values of inputs are calculated through input membership functions and outputs are calculated through output membership functions.

X axis of this graph is the input value. Y value of the graph is the membership value of the fuzzy sets. For example: according to the graph, if we choose value 2 for the input, it is member of the "High" fuzzy set and the membership value of the input is 1 (Fig. 4).

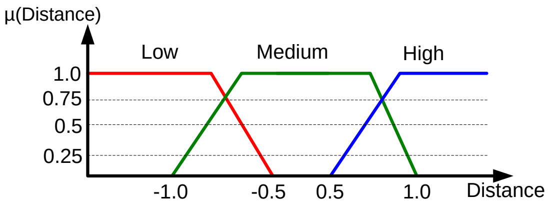

Fig. 4. Input membership fuzzy sets.

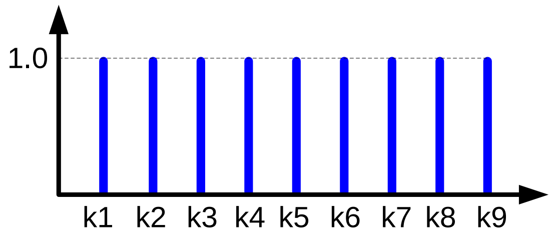

Fig. 5. Output membership functions.

The following set of rules shown in Table I are used for the enemy drones:

TABLE I. RULES IN THE FUZZY DRONE CONTROL

X Distance	Y Distance	Consequent
Left	Middle	West (k1)
Left	Top	North West (k2)
Middle	Top	North (k3)
Middle	Middle	Explosion (k4)
Right	Top	North East (k5)
Right	Middle	East (k6)
Right	Bottom	South East (k7)
Middle	Bottom	South (k8)
Left	Bottom	South West (k9)

These rules are checked from the rule base, and the membership values are sent back to the inference engine. Inference engine gets output membership values and takes the output membership values and places them into an integer array. Average of these values are taken and placed to output membership graph of Fig. 5. The average crisp value is placed at the X axis of the graph. The closest output function of that value is chosen as the final output function. Then the enemy drone shoots itself according to the output function.

Every enemy drone has its own inference engine. Because if there are more than one drone trying to get calculations from inference engine there will be conflict problems. Individual inference engine solves this problem.

Let's see an example how an enemy drone detects the user spaceship and fires itself to the chosen direction.

If we look at Fig. 6 we can see the movement of enemy drone. At the moment of enemy drone radar scan, user spaceship is at the north east of the enemy drone. X and Y distances are both 2 units. If we look at the Fig. 4 we can calculate the membership values of these crisp inputs. These are both equivalent to 1 and both are members of fuzzy set "High". Then inference engine sends these linguistic variables with their membership values and a specific rule (Table I) is fired:

```
If X is "High" and Y is "High" then
direction is "North East".
```

North east is the fifth output membership function where enemy drone shoots itself to the north east direction.

Obviously there is only 1 rule fired and average of the output membership function values is same as the output of fifth rule. But if there were more than one rule fired, then there would be multiple outputs. In this case, the average of all these values would be chosen and the closest output function would be chosen as the crisp output value.

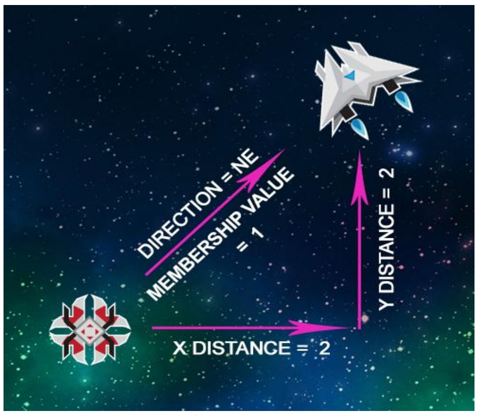

Fig. 6. Example of an enemy drone movement.

VI. Result

During the calculations and tests on gameplay, enemy drones hit user spaceship mostly accurately. Some of the times user spaceship moves too fast and when enemy drone fires itself towards the chosen direction, speed of the enemy drone is not fast enough to catch the user spaceship. But this is a very rare situation, user spaceship doesn't move too fast most of the times.

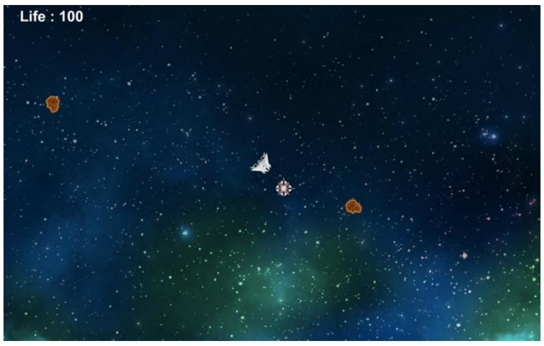

Fig. 7. Drone approaching the spaceship.

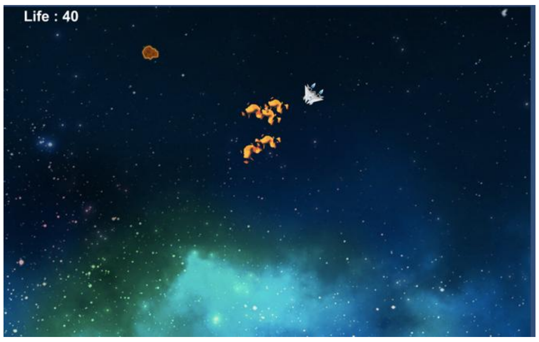

Fig. 8. Explosion effect after the drone collides.

During the implementation of fuzzy logic there were some problems. At first only the membership values were taken as the output value evaluation. This resulted in concluding for the incorrect directions for motion. Every rule fired produces a membership value, and this value must be multiplied with a constant value declared before. Sugeno output functions are singletons, and every membership value sent to the functions are multiplied with a constant variable. Average of all these values are chosen as the output value as a result enemy drone can shoot itself to the right direction more frequently as shown in Figs. 7 and 8.

VII. Conclusion

Fuzzy logic is a powerful tool for the AI implementations inside the games. Of course state machines would work without any problem for such games. However, bigger and more complicated AI systems can have problems with the state machines. Too many options can produce too many state machines, which makes it harder for the developers. Making a minor change inside this AI system will also be very hard because of the quantity of the states.

When fuzzy logic is employed, every component can be separated from each other and can be manipulated easily. Addition or removal of rules from the rule base is much faster and can be done even without a high level of programming experience. Also the behaviours of the AI characters inside the game are connected to each other with fuzzy inference. For instance, a character can have emotions, physical strength and movement systems. With fuzzy inference system AI characters can decide where to move or what to do according to emotional state and physical awareness. This makes AI actions more realistic.

In this paper we have shown that a fuzzy logic based AI implementations can make significant contributions to game development as well as making the game more challenging and hence enhancing the gameplay. The enemy drones operated with zero order Sugeno inference engine can decide on the time for attacking the spaceship and giving damage. The player would not be able to guess whether a drone is in sleep mode (not sending radar signals) or is ready to attack.

For research purposes, fuzzy logic should be used more commonly inside AI systems for games since it is a quite efficient way to see AI behaviour implemented characters inside games.

References

[1] M. Buckland, *Programming Game AI by Example*, Jones& Bartlett Learning, 2005.

[2] M. Negnevitsky, *Artificial Intelligence: A guide to Intelligent Systems*, Addison-Wesley, Reading, MA, 2005.

[3] M. Pirovano, "The use of fuzzy logic for artificial intelligence in games," Department of Computer Science, University of Milano, Milano, Italy, 2012.

[4] D. Johnson and J. Wiles, "Computer games with intelligence," *Proceedings of the FUZZ-IEEE*, pp. 1355-1358, 2001.

[5] S. Woodcock. (August, 1999). Game AI: The State of the Industry. [Online]. Available: http://www.gamasutra.com/features/19990820/game_ ai_01.htm

[6] S. Woodcock. (2008). Games making interesting use of artificial intelligence techniques. [Online]. Available: http://www.gameai.com/games.html

[7] P. Sweetser, "Emergence in games," *Cengage Learning*, 2008.

[8] T. Ross, *Fuzzy Logic with Engineering Applications*, Wiley, 1999.

[9] L. A. Zadeh, "Fuzzy logic = computing with words," Computer Science Division/Electronics Research Laboratory, Department of EECS, University of California, 1996.

[10] Anonymous. (2016) Unity Tutorials. [Online]. Available: https://unity3d.com/learn/tutorials

Ali Emre Soylucicek is currently a senior student at Ankara University, Computer Engineering Department, Turkey. He graduated from Golbasi Anatolian High School. He completed Erasmus+ exchange program in Poland – Krakow for one year as well as participating in Erasmus+ internship program in Krakow. He performed his internship at a game company called "Duckie Deck". He is currently working part-time GelişimPark Inc. at Cyberpark-Bilkent.

He is focused on developing games as a programmer for both mobile and other platforms. He also conducted research about artificial intelligence implementations inside games. He improved himself by doing researches about Unity3D game engine. He created variety of games on both iOS and Android games, also published them to the market. He is currently developing more games and doing research in game development and artificial intelligence implementations inside games.

Erkan Bostanci received a BSc degree in the Computer Engineering Department from Ankara University, Turkey in 2007. Consequently, he joined the same department as a Research Assistant and completed his MSc on real-time battlefield simulation in 2009. He obtained his PhD from School of Computer Science and Electronic Engineering, University of Essex, United Kingdom in 2014 with his thesis on real-time user tracking for augmented reality.

He started working with the Gendarmarie Schools Command as a Planning Officer Designate in June, 2014 where he conducted the research for developing a vision-based system for analysing crime scenes. He has been promoted to Second Lieutenant in January, 2015. Having completed his military service, he currently continues his post in Ankara University as an Assistant Professor.

His research interests include different yet closely related aspects of computer science from image processing, computer vision and graphics to artificial intelligence and fuzzy logic as well as mathematical modelling and statistical analysis. He recently developed a vision-based user tracking system for various augmented reality applications for cultural heritage in particular. He setup the SAAT laboratory in the department for conducting research with the main aim for incorporating AI approaches for solving a wide range of real world problems.

Dr. Bostanci has been involved in technical committees for several conferences and acted as a reviewer for various journals.

Aykut Burak Safak is a senior student in Ankara University, Computer Science Department. He graduated from Bor Akin Gonen Anatolian High School. He completed two summer internships at TaleWorlds Entertainment. His work interests include computer graphics, artificial intelligence, genetic algorithms and fuzzy logic. He has been working in TaleWorlds Entertainment as a junior developer since 2014.

He is focused on developing artificial intelligence in games. His studies also focus on the graphics programming in games. He created several games using Unity 3D and SFML and a simple game engine using Open-GL for the purpose of self-development yet haven't published any of them.

Computer Science and Applied Technology