# THE GRAND N.S.M. WAR: A FUZZY GAME SIMULATION OF MULTI-ARMY MULTI-AGENT ACTIVITIES

*ANJALI SINHA(A0178476L), GOPALAKRISHNAN SAISUBRAMANIAM(A0178249N), KENNETH RITHVIK(A0178448M), MADAN KUMAR MANJUNATH(A0178237W), VIKNESH KUMAR BALAKRISHNAN(A0178304E)*

Institute of Systems Science, National University of Singapore

**ABSTRACT**

The objective of the project is to empower game development with an illusion of intelligence. Fuzzy logic is one among many techniques opted as a choice because of its simplicity in its implementation along with its expressive power. We discuss the design and implementation of a fuzzy-based AI system for a novel multi-army, multi-agent war game. The game has been modeled to resemble an automated simulation of activities executed by the developed characters - Ninja, Samurai and Mage, which would attack and battle out against each other based on each character's unique fuzzy intelligence. We discuss the different components of the game, variables, fuzzy rules, short comings and the future enhancements in the upcoming sections.

## 1. INTRODUCTION

The audio and visual aspect of games have improved a lot over the years, but AI makes way for even more improvisation. Fuzzy logic has been commonly applied in games as seen in [1] [2] Humans can adapt to their environment, but characters in the game follow certain predefined flows and paths. The addition of fuzzy terms and rules make it more realistic and simpler to control. A multi-agent game is built, where we try to leverage fuzzy logic to enable a swarm of agents to make a collective decision against other counterparts in vicinity. We came up with a set of fuzzy variables and rules to give each agent the sense of human-like decision making.

## 2. BACKGROUND

### 2.1. Fuzzy Inference Systems

Fuzzy logic provides approximation methods for dealing with information that does not have well-defined boundaries. Humans can effectively deal with the information with imprecision and vague boundaries whereas machines are more suited to arrive at decisions based on truth values and constants. Fuzzy logic minimizes the gap so that it becomes easier to express human ideologies to a machine to complete an activity.

Fuzzy Inference System is the key unit of a fuzzy logic system and its primary work is decision making. It uses the "IF…THEN" rules along with connectors "OR" or "AND" for drawing essential decision rules.

Fuzzy rule inference is a three-step process:

1. **Fuzzification**: determining the membership value of the crisp input in the different fuzzy sets of linguistic variables
2. **Fuzzy rules**: apply fuzzy rules to the fuzzified input in order to determine the fuzzy output.
3. **Defuzzification**: usually fuzzy output is converted back to a crisp value, especially for game, to be used in the application

There are 2 kinds of fuzzy inference systems; Mamdani-type and Sugeno-type. Mamdani's fuzzy inference method is the most commonly seen fuzzy methodology and is the default in skfuzzy package. [3] [4]

### 2.2. Pygame for UI design

Pygame is a cross-platform set of Python modules designed for creating video games. It provides support for implementing computer graphics and sound libraries designed to be used with the Python programming language. Pygame uses the Simple Direct Media Layer (SDL) library, with the intention of allowing real-time computer game development without the low-level mechanics of the C programming language and its derivatives. [5]

## 3. THE GAME

N.S.M. war is a multi-agent battle game, where three armies - Ninjas, Samurais and the Mages fight each other to remain the last one's standing. The game has an enclosed battlefield within which all the characters(also referred to as agents) fight. It also has high walls in the battlefield, which act as obstacles and limit mobility of the agents. Each of the 3

agents have their own set of properties in terms of speed, health and movement decisions. During the game play, several parameters such as attack rate, speed, agent count etc. can be put in by the user. The game then simulates a battle based on several fuzzy parameters as well as game conditions combined. Each agent makes its own decisions.

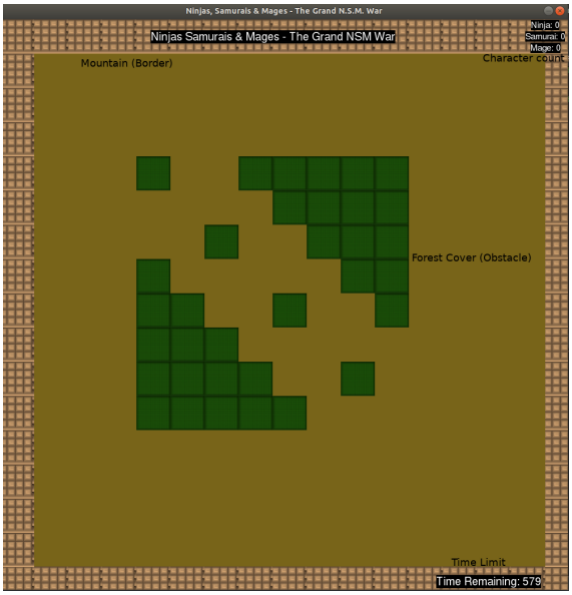

Fig.3.) The battleground

### 3.1. Design Overview

In this game all the characters are intelligent agents, and human intervention is required only before the game to set configurable parameters to simulate the battlefield. The agents derive conditions from the environments to make their decisions. To make the game more realistic, each kind of character has varying sight and mobility (in terms of steps taken). But the mobility can change if the conditions in the game become unfavorable.

By default, Ninjas are fast and scan a wider range, but have lower attack power.

Fig.3.1.1.) Ninja

Samurais have a balance between attack and speed but are limited by their scan range.

Fig.3.1.2) Samurai army showing the scan range

Mages are the strongest among the three and have decent scan range. However their movements are slow and they have to recharge longer as their spells drain a lot of magic.

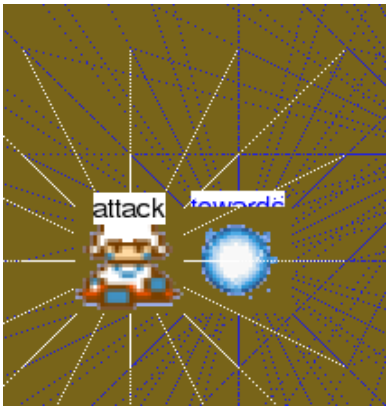

Fig.3.1.2) Mage casting powerful energy ball towards opponent

Each agent has Health associated to it. If an enemy attacks, the agent will get hit which will reduce its health. Once the health is zero, the agent is “dead” and does not contribute to the battle anymore.

The game environment is developed in such a way that it can be modified, and a completely new arena can be built. For example, the forest cover radius can be configured which dynamically creates trees in the ground thus serving as obstacles.

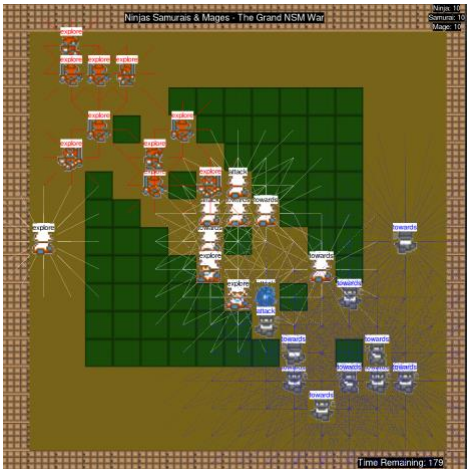

Fig.3.1.3.) Modifying the forest cover (obstacle)

### 3.2. Implementation

The user can configure the following parameters before running the simulation:

1. Ninja Army Count
2. Samurai Army Count
3. Mage Army Count
4. Ninja
    a. Speed
    b. Attack
    c. Scan_Range
    d. Max_Charge
    e. Start_Position
    f. Fuzzy_Roam
5. Samurai
    a. Speed
    b. Attack
    c. Scan_Range
    d. Max_Charge
    e. Start_Position
6. Mage
    a. Speed
    b. Attack
    c. Scan_Range
    d. Max_Charge
    e. Start_Position
7. Game Configuration
    a. Frame_Rate
    b. Window_Width
    c. Window_Height
    d. Seconds
    e. Forest_Radius

Different combination can boost or weaken an army and ultimately affect the war's pace and winners.

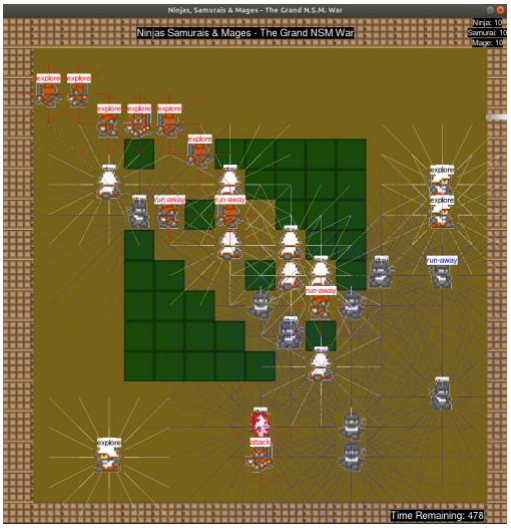

Fig.3.2.1.) Initial screenshot of waging war

To study the influence of individual characters, there is an option called 'No War', a flag that prevents the game ending in the event of no enemies.

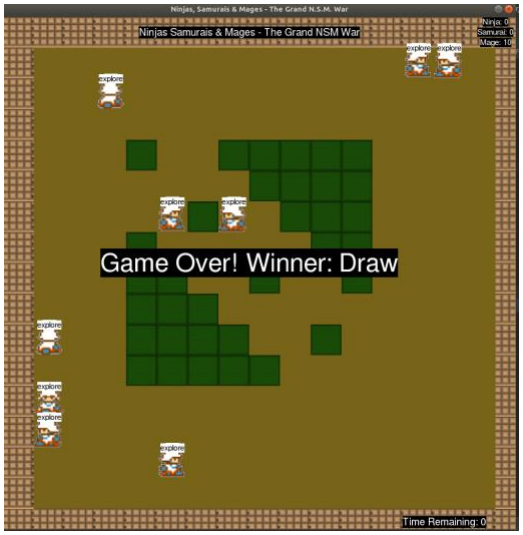

Fig.3.2.1.) No War simulation – Draw after time runs out

## 4. FUZZY RULES & GAME BEHAVIOR

### 4.1. Fuzzy Rules

The agents need to perform intelligent actions, hence 3 different modes of decision-making are chosen. Agents can chose to either *move*, *stay* in place or *attack*.

The first decision that has been implemented is "stay". The agent here stays in the same position and doesn't move unless the next decision making cycle. The second decision implemented is to "move". Depending on the scanning range, each agent can chose to either move "towards" or

"away" from other agents, or "explore". The third decision is to "attack". This depends on the army size of enemies, as well as size of agent's own army in the scanning range. If the agent's army have an upper hand(in terms of size), the agent makes an individual decision to attack the enemy agents.

Several fuzzy rules contribute to this decision-making for the characters in the game. The inputs are linguistic terms that are obtained from the agent's health and proximity of enemy army and the agent's own army.

The following rules are responsible for "stay" decision of the agents:

- If (ninja_team = *small* and ninja_enemy_samurai = *small*) Then *stay*
- If (mage_team = *one* and mage_enemy_ninja = *medium*) Then *stay*
- If (samurai_team = *small* and samurai_enemy_ninja = *small*) Then *stay*
- If (mage_team = *small* and mage_enemy_samurai = *medium*) Then *stay*
- If (mage_team = *small* and mage_enemy_ninja = *large*) Then *stay*
- If (ninja_team = *medium* and ninja_enemy_samurai = *large*) Then *stay*

The following rules are responsible for "move" decision of the agents:

- If (ninja_team = *small* and ninja_enemy_samurai = *medium*) Then *move*
- If (ninja_team = *one* and ninja_enemy_mage = *large*) Then *move*
- If (samurai_team = *one* and samurai_enemy_mage = *large*) Then *move*
- If (samurai_team = *medium* and samurai_enemy_ninja = *large*) Then *move*
- If (mage_team = *one* and mage_enemy_ninja = *medium*) Then *move*
- If (mage_team = *small* and mage_enemy_samurai = *large*) Then *move*

The following rules are responsible for "attack" decision of the agents:

- If (ninja_team = *one* and ninja_enemy_samurai = *one*) Then *attack*
- If (ninja_team = *large* and ninja_enemy_mage = *small*) Then *attack*
- If (samurai_team = *one* and samurai_enemy_ninja = *one*) Then *attack*
- If (samurai_team = *medium* and samurai_enemy_mage = *small*) Then *attack*
- If (mage_team = *one* and mage_enemy_ninja = *small*) Then *attack*
- If (mage_team = *large* and mage_enemy_samurai = *small*) Then *attack*

### 4.2. Neighborhood Scan

The three kinds of agents have a "scan area" feature, with the following ranges set as default, and configurable in the game –

1. Ninja's scanning area is a 9×9 matrix
2. Samurai's scanning area is a 7×7 matrix
3. Mage's scanning area is a 5×5 matrix

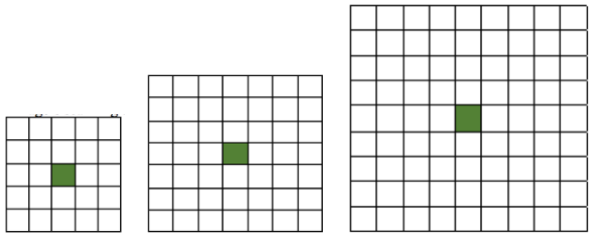


Fig.) Scan Area for (a) Mage, (b) Samurai, (c) Ninja. The green square represents the position of the respective agent type.

This scanning area represents the visibility range for each of the characters. A character is only able to see the any other agents who are within the scan area.

### 4.3. Collective Decision Making

The rules clubbed along with the scan range, helps determine the direction of movement and speed of the agent. At each decision making cycle, an agent will scan the area around itself, get the count of similar agents as well as the enemy agents. Pass it into the fuzzy model to get linguistic groups which will again be passed into another fuzzy model to get the decision for the agent.

Eg: A ninja(In the green square) will scan the 9×9 grid. The ninja's visibility range say that including himself, there are 8 ninja's(maps to fuzzy team size = large), 2 Samurai's (fuzzy team size = small) and 3 Mage's (fuzzy team size = medium). This triggers rules and comes the decision to attack the samurai army in the range. In order attack, it will make a non-fuzzy decision to make a move *towards* the enemy before annihilating them. While moving towards the enemy, the speed becomes slower, to ensure that it reaches the enemy.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | N | | | | | | N | |
| | | | | | | | | |
| N | | N | N | | | | | |
| | | | | | | | | |
| | | | | N | N | | | |
| | | | | | | | | |
| | M | | | | | | | |
| | | | | | S | S | | |
| M | M | | | | | | | N |

If the agent comes to the decision that the enemy size is large, it expects an attack and hence, moves “away” from the enemy with an increased speed. This ensures the agent’s escape for that decision cycle.

## 5. FUZZY LINGUISTIC TERMS

A fuzzy linguistic variable is a collection of fuzzy set which represents a degree of a concept [6] Not all variables can be defined with crisp values. In order to have a intuitive way of representation for the humans we can use fuzzy linguistic terms to represent a variable. This would also be easier for a knowledge engineer give information on a variable behavior. Our game also has variables which are described with fuzzy terms. It would further ease in framing the rules.

### 5.1. Speed Variable

Depending on the magnitude of the speed fuzzy sets are defined for this variable. It encapsulates fuzzy sets slow, medium and fast. Further rules are built on this variable using these fuzzy sets. Trapezoidal membership function is used to define this function.

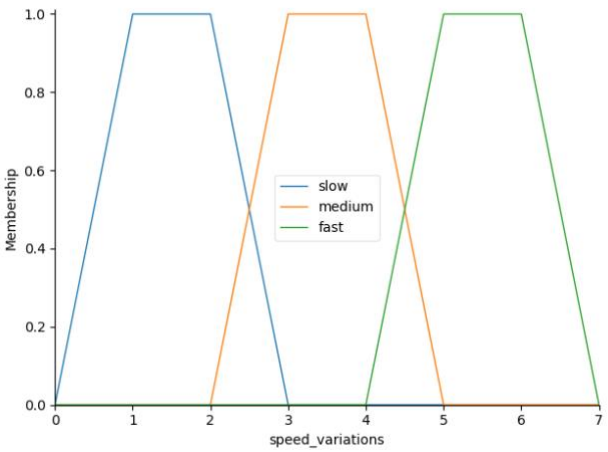


Fig.5.1.) Membership function for Speed

### 5.2. Attack and Health variable

The characters of our game have the attacking power to battle with the opponents. Also, each character has a health index. Once all the health is lost the character dies in the game These variables can be expressed in fuzzy sets. Low, Medium and High are the fuzzy linguistic terms used to describe the magnitude of attack and the health index by the characters. We have used triangular membership function to define this variable

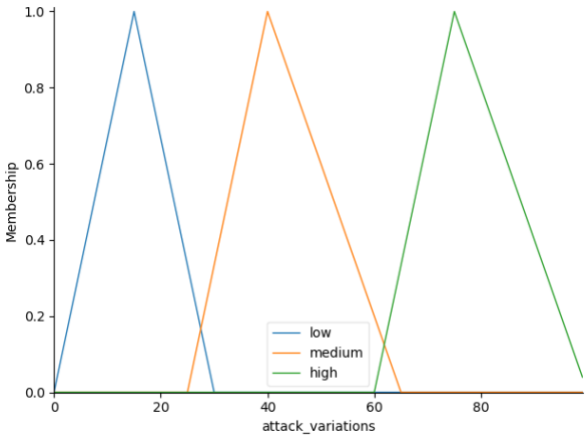


Fig.5.2.) Membership function for Attack

### 5.3. Movement variable

Depending on the positions of the opponents in the surrounding area the characters need to make a movement decision. ‘Stay’, ‘Attack’ and ‘Move’ are the fuzzy sets for movement variable. Trapezoidal membership function is used to define this function.

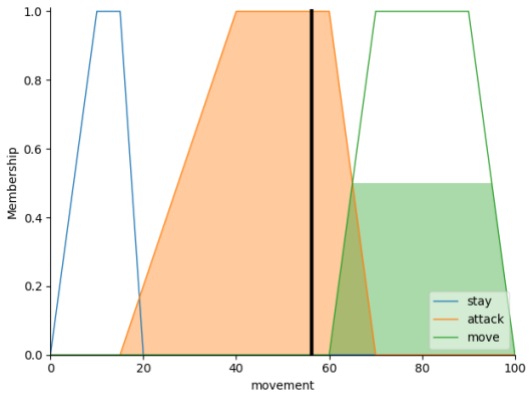


Fig.5.3.) Membership function for Movement

### 5.4. Roam variable

The Ninja character have a configurable variable which determines whether to fight with a team strategy. It is defined with a fuzzy set ‘Yes’ and ‘No’. We have used triangular membership function to define this variable. The wide coverage of yes is intended.

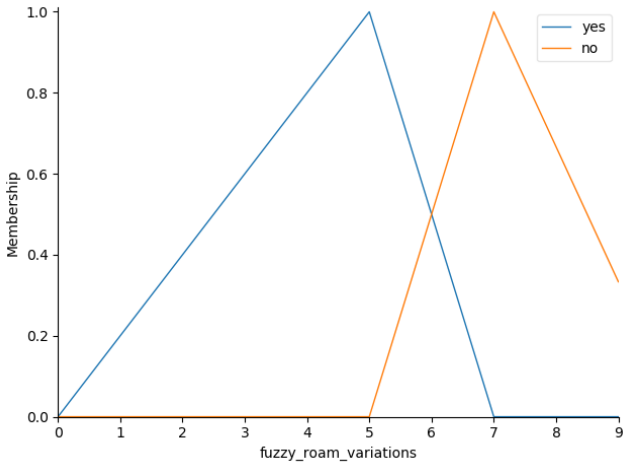


Fig.5.4.1.) Membership function for Fuzzy_Roam

The below screenshot shows the visual simulation of fuzzy roam.

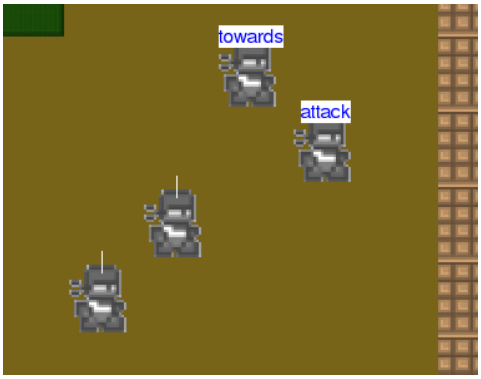


Fig.5.4.2.) Ninja Fuzzy Roam

It has been observed that the Fuzzy Roam ability works well for Ninjas and they have been successful in attacking together or staying out of battle till the end, thus leading to their victory in several trials.


Fig.5.4.3.) Ninjas win the war

### 5.5. Team Variables

In the battle field we have three different army fighting against each other. They are Ninja, Samurai and Mage. Based on the number of people in each army the fuzzy terms are defined for each of the team. The fuzzy term set include

i) One – Only one member in the army
ii) Small – More than one but still small in number
iii) Medium – Having an average count of army members
iv) Large – A relatively higher count than the average range of the defined function.

The member ship functions for the respective teams are defined as below. Trapezoidal membership function is used to define these functions.

#### 5.5.1. Samurai Army

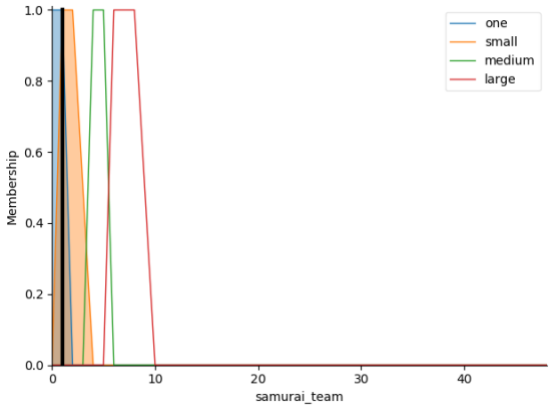


Fig.5.5.1.) Membership function for Samurai Army Size

#### 5.5.2. Ninja Army

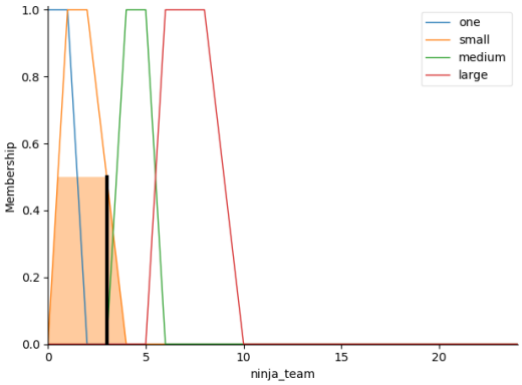


Fig.5.5.2.) Membership function for Ninja Army Size

### 5.5.3. Mage Army

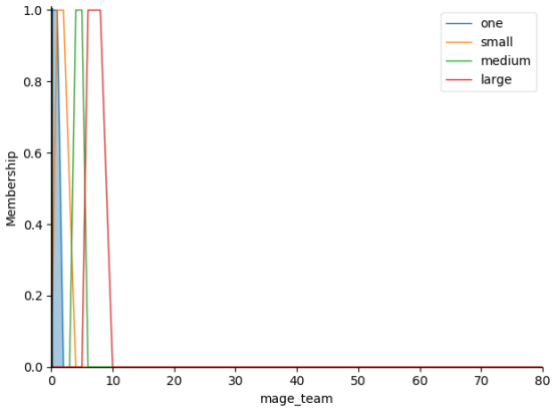


Fig.5.5.3.) Membership function for Mage Army Size

## 5.6. Enemy Variables

As it's a three army battle each character will have a probablity of encountering a co-army member or any one of the character from either of the two different type of enemies. When it encounters a co-army member they need not perform any movement. On the other hand depending on the enemy army strength they have to make necessary movement plans. The fuzzy set in this case are also set as follows:-

i) **One** – Only one member in the enemy army
ii) **Small** – More than one but still small in number
iii) **Medium** – Having an average count in enemy army
iv) **Large** – A relatively higher count than the average range of the defined function.

As each army might be encountering with two other different army we have two more membership functions defined for each army. Trapezoidal membership function is used to define these functions.

### 5.6.1. Samurai v/s Enemy Ninja

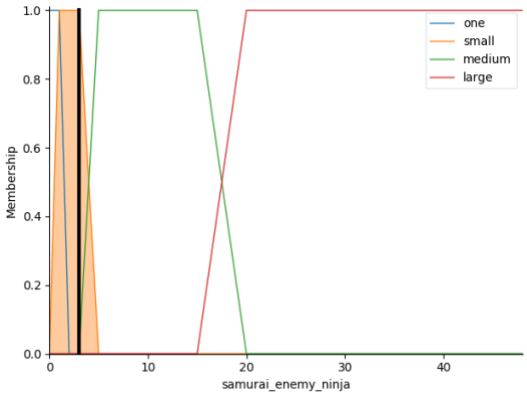


Fig.5.6.1.) Membership function for Samurai vs Enemy Ninja

### 5.6.2. Samurai v/s Enemy Mage

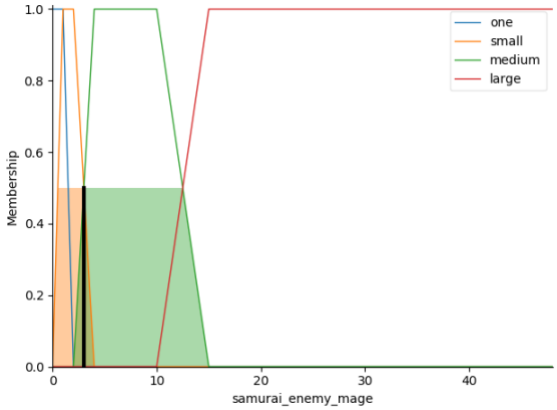


Fig.5.6.2.) Membership function for Samurai vs Enemy Mage

### 5.6.3. Mage v/s Enemy Ninja

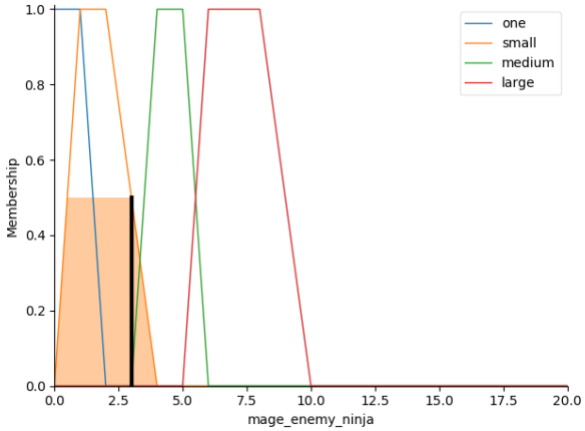


Fig.5.6.3.) Membership function for Mage vs Enemy Ninja

### 5.6.4. Mage v/s Enemy Samurai

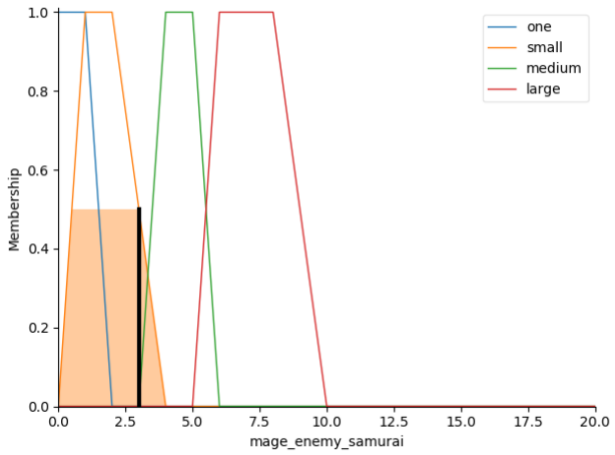


Fig.5.6.4.) Membership function for Mage vs Enemy Samurai

### 5.6.5. Ninja v/s Enemy Samurai

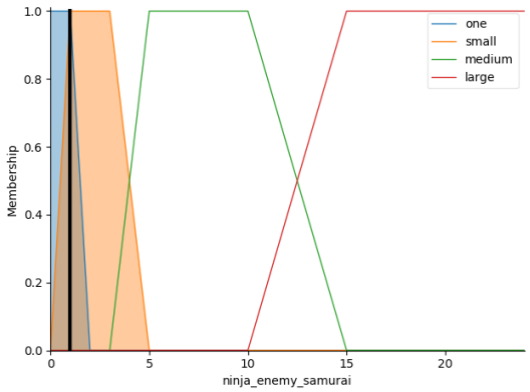


Fig.5.6.5.) Membership function for Ninja vs Enemy Samurai

### 5.6.6. Ninja v/s Enemy Mage

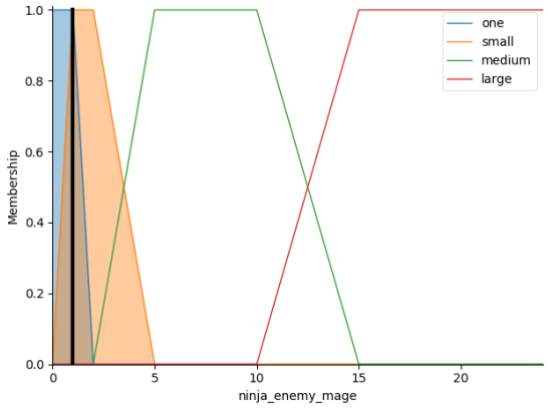


Fig.5.6.6.) Membership function for Ninja vs Enemy Mage

## 6. FUZZY RULE INFERENCE & PARAMETER SELECTION

To get a single decision for an agent from fuzzy calculations, we need to defuzzify the outputs. The value of the output is calculated using a series of steps. The outputs which relate to the same decision making are combined using center of average method. The defuzzified output for decision making is in the range [0,100].

Once the defuzzified outputs have been obtained, we use the following function to make a decision:-

```
# return either of 3 choices
    if (decision <= 10):
        return "stay"
    else if (decision >30 and decision <=70):
        return "attack"
    else:
        return "move"
```

Below are some screenshots of results taken at different scenarios:

```
**********
FUZZY INPUTS:
health: 92
goblin_team: 1
ogre_team: 0
troll_team: 1
defuzzified decision value: 60.76923076923078
**********
```

Fig.6.1.) Sample Fuzzy Input and corresponding de-fuzzified output


Fig.6.2.) Decisions and actions made by agents


Fig.6.3.) Samurai wins the war

## 7. FUTURE WORK

This is the first version of the NSM Wars, and plan to have several additions to this game in the next phase. We plan to incorporate reinforcement learning along with the fuzzy logic, to make the agents learn from their experiences in the battlefield simulation.

Another idea is to mimic closer to human nature - although certain agents in the game seem to be based on the fuzzy rules set by the user, there can be still some unrealistic behavior observed, which has not been taken care of in this project. If same army agent dies beside an agent, the agent would not make a move based on that. In a player-controlled game that would be a cue to move away from that place.

We also plan to make changes to the UI. Adding a health bar for each character, so that the viewer can actually keep a track of it through UI, instead of the logs. We also plan to make the game 3D, using OpenGL.

## 8. CONCLUSIONS

A simulation of multi-army multi-agent game was created and given intelligence using Fuzzy logic concepts. Different rules were created and their composition demonstrates realistic actions which makes the game enjoyable. Tuning different parameters in the configuration provides scope for different battles and yields different results.

## REFERENCES

[1] PacMan : http://www.ccis2k.org/iajit/PDF/vol.3,no.4/7-Adnan.pdf
[2] Fuzzy logic based game design : https://ieeexplore.ieee.org/document/1337393
[3] skfuzzy https://pythonhosted.org/scikit-fuzzy/
[4] skfuzzy compute rule : https://pythonhosted.org/scikit-fuzzy/_modules/skfuzzy/control/controlsystem.html#ControlSystemSimulation.compute_rule .
[5] Pygame : https://www.pygame.org/docs/
[6] Fuzzy logic for games : http://purvag.comblog/?p=284