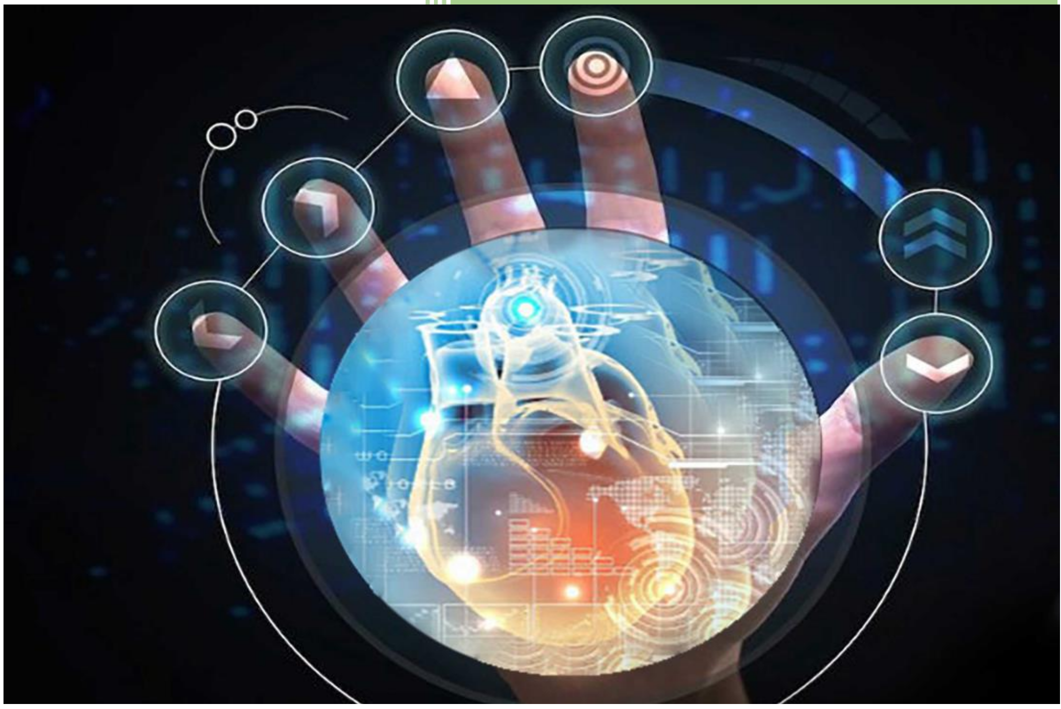
2019

Readme



Naqi Ahmad - STUDENT

Contents

Project Description

Controlling Game Characters with a Neural Network and Fuzzy Logic.

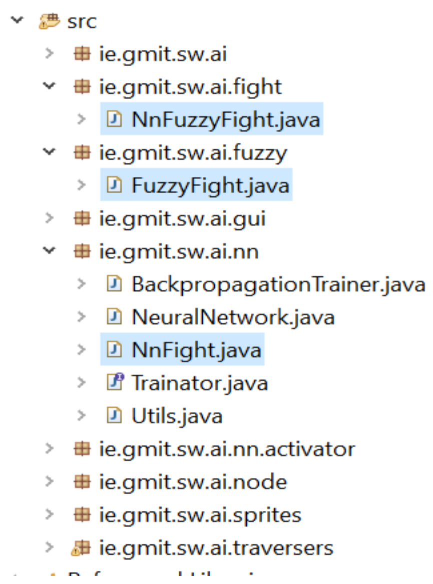
We were required to create an AI controlled maze game containing the set of features. The objective of the game is to escape from a maze and either avoid or fight off the enemy characters that move around in the game environment.

Threads

The player and enemy sprites execute in separate threads. I have the player and the Spider running on separate threads. Searches are performed in threads by each node individually.

Fuzzy Logic

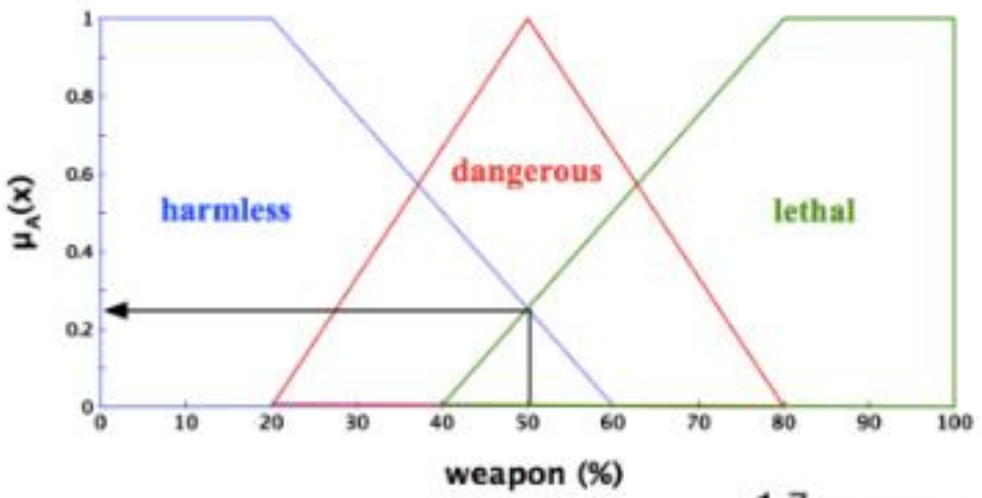
In the resource folder I have defined the sets and linguistic variables for my fuzzy logic. I use Fuzzy Logic to calculate the damage when the spider and player Combats. The players health is determined by a fuzzy logic classifier that will return damage done by the spider. Whenever the player goes near the spider, he loses health depending on what type of spider it is. The spider checks for the player using the Heuristic searches. When the spider finds the player, he attacks him and causes the player to lose the health.



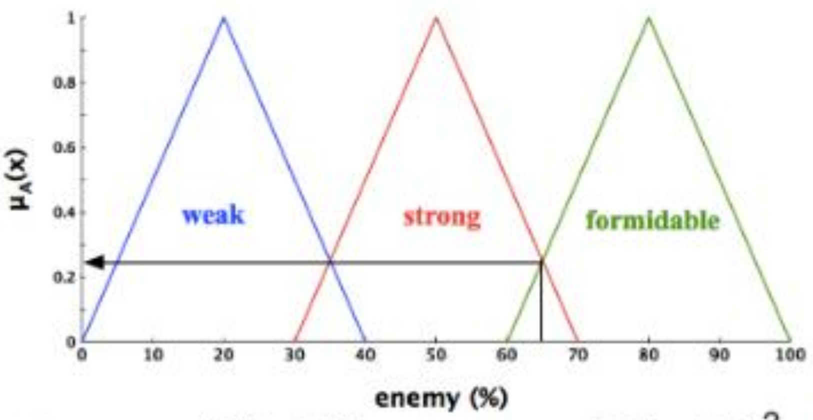
```
src
  ie.gmit.sw.ai
  ie.gmit.sw.ai.fight
    NnFuzzyFight.java
  ie.gmit.sw.ai.fuzzy
    FuzzyFight.java
  ie.gmit.sw.ai.gui
  ie.gmit.sw.ai.nn
    BackpropagationTrainer.java
    NeuralNetwork.java
    NnFight.java
    Trainator.java
    Utils.java
  ie.gmit.sw.ai.nn.activator
  ie.gmit.sw.ai.node
  ie.gmit.sw.ai.sprites
  ie.gmit.sw.ai.traversers
```

Fuzzy Sets

```
-------------------Input-----------------------------
// depending on the weapon of spider the greater the attack would be
FUZZIFY weapon
  TERM harmless := (0, 1) (20, 1) (60, 0);
  TERM dangerous := trian 20 50 80;
  TERM lethal := (40, 0) (80, 1) (100, 1);
END_FUZZIFY

// if the enemy strenth is between 0 and 40 its going to be weak
// if the its between 30 and 70 its going to be in the strong range
// if the enemy is between 60 and 100 its going to be formiddable
// the stronger the enemy the greater the attack is going to be
FUZZIFY enemy
 TERM weak := trian 0 20 40;
 TERM strong := trian 30 50 70;
 TERM formidable := trian 60 80 100;
END_FUZZIFY
```

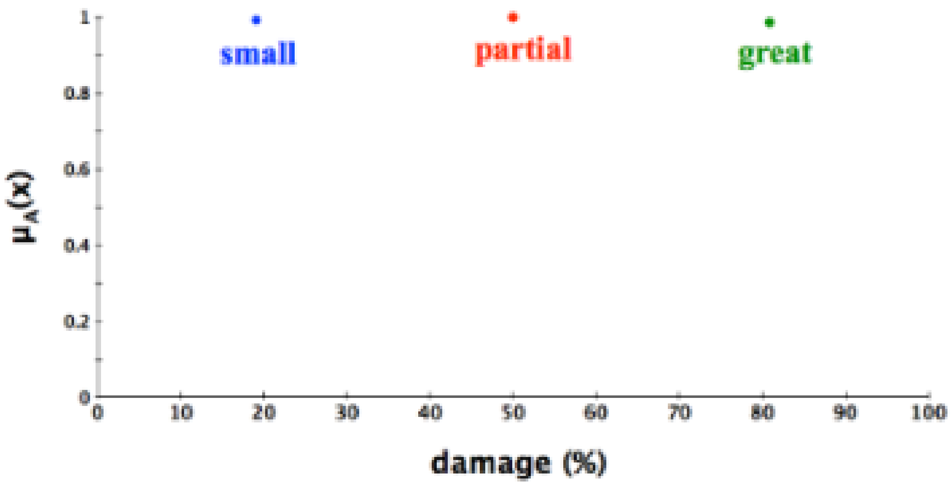


$\mu_{lethal}(50) = 0.25, \therefore \mu_{slightly}(0.25) = 0.25^{1.7} = 0.0947$



$\mu_{formidable}(65) = 0.25, \therefore \mu_{extremely}(0.25) = 0.25^{3} = 0.125$

```
-------------------Output-----------------------------
//here i calculate the damage which is going to be ouput to determine the new health
// which will be used to calculate the new health of the player
DEFUZZIFY damage
 TERM small := (19,0) (20, 1) (21, 0);
 TERM partial := (49,0) (50, 1) (51, 0);
 TERM great := (79,0) (80, 1) (81, 0);
 METHOD : COG;  // improves the control performance of a fuzzy logic controller
 DEFAULT := 0;
END_DEFUZZIFY
```



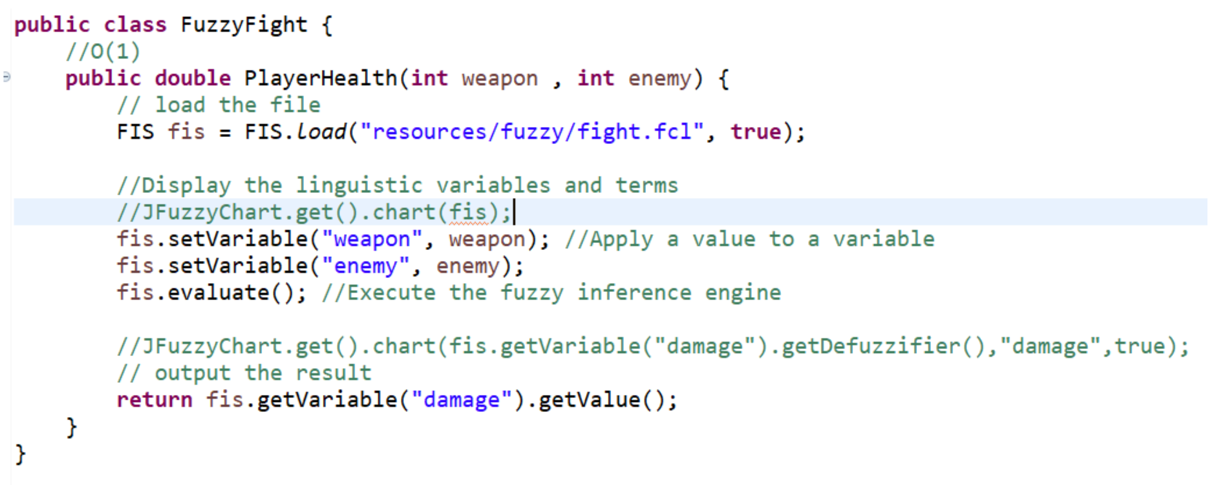
$$\mu_{great}(x) = \min(0.0947, 0.125) = 0.0947$$

Fuzzy Rules

```
-------------------Rules----------------------------
// RULE 1 is basically if the spider weapon is dangerous or enemy is strong then its going to damage players health partially
// RULE 2 if the spider weapon is lethal or enemy is formidable then the risk is also going to be partial
// RULE 3 if the spider weapon is harmless or enemy is weak then its going to be low damage to the player
// RULE 4 if the spider weapon is dangerous and enemy is strong then its going to damage players health it partially
// RULE 5 if the spider weapon is lethal and enemy is formidable then damage to players health is going to be great
// RULE 6 if the spider weapon is harmless and enemy is weak then its going to be low damage to the players health
RULE 1 : IF weapon IS dangerous OR enemy IS strong THEN damage IS partial;
RULE 2 : IF weapon IS lethal OR enemy IS formidable THEN damage IS partial;
RULE 3 : IF weapon IS harmless OR enemy IS weak THEN damage IS small;
RULE 4 : IF weapon IS dangerous AND enemy IS strong THEN damage IS partial;
RULE 5 : IF weapon IS lethal AND enemy IS formidable THEN damage IS great;
RULE 6 : IF weapon IS harmless AND enemy IS weak THEN damage IS small;
```

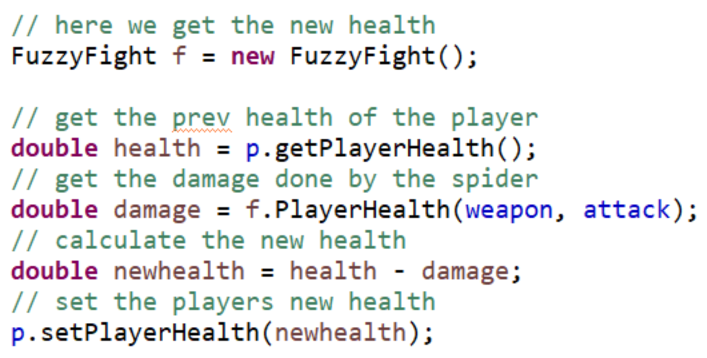
Game Integration

This is how I am using the fuzzy logic. It returns the damage that the player incurs.



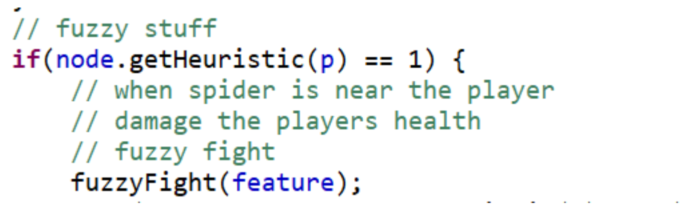
```
public class FuzzyFight {
    //O(1)
    public double PlayerHealth(int weapon , int enemy) {
        // load the file
        FIS fis = FIS.load("resources/fuzzy/fight.fcl", true);

        //Display the linguistic variables and terms
        //JFuzzyChart.get().chart(fis);
        fis.setVariable("weapon", weapon); //Apply a value to a variable
        fis.setVariable("enemy", enemy);
        fis.evaluate(); //Execute the fuzzy inference engine

        //JFuzzyChart.get().chart(fis.getVariable("damage").getDefuzzifier(),"damage",true);
        // output the result
        return fis.getVariable("damage").getValue();
    }
}
```

In class NnFuzzyFight.java I use the above class to attack the spider and get new health.



```
// here we get the new health
FuzzyFight f = new FuzzyFight();

// get the prev health of the player
double health = p.getPlayerHealth();
// get the damage done by the spider
double damage = f.PlayerHealth(weapon, attack);
// calculate the new health
double newhealth = health - damage;
// set the players new health
p.setPlayerHealth(newhealth);
```

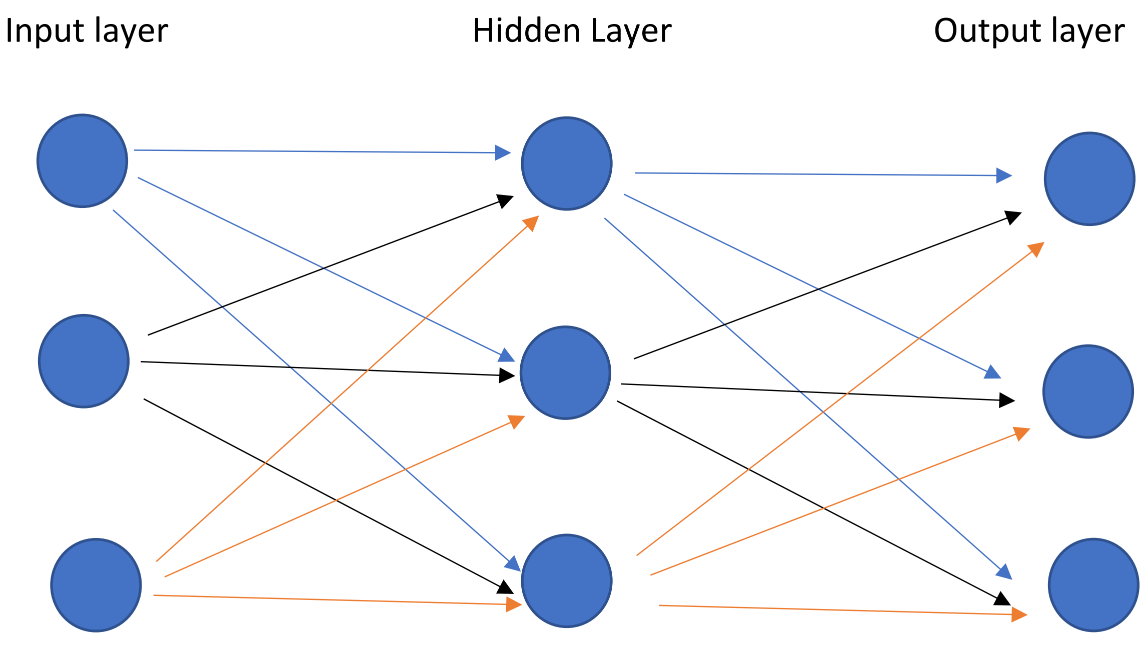
In my run method I check if the player is close to spider and it is then the spider will attack and reduce players health.



```
// fuzzy stuff
if(node.getHeuristic(p) == 1) {
    // when spider is near the player
    // damage the players health
    // fuzzy fight
    fuzzyFight(feature);
```

Neural Network

I created a very extensive data set for the spider to help the decision making either to panic, attack or hide from the player depending on what players health, weapon and strength of the spider are. I am using the Sigmoid Network topology. The reason I chose the sigmoid was because it is easily differentiable with respect to the network parameters. This type of Neural network is also very easy to train. The Neural net is trained at the start of the game in the Game Runner and passed to the maze and from there I use it to pass it to the NnFuzzyFight class. This way I don't have to train the neural network every time spider is close to the player. This is done to ensure that the game is not slow.



Layers

- The first layer takes in 3 inputs which are the players health, Players weapon and spider strength.
- I chose 3 hidden layers because I found it to be the best option from trial and error. Used the method below to see how many hidden nodes would be good.

$$N_h = \frac{N_s}{\alpha \cdot (N_i + N_0)}$$

N_h = number of hidden neurons.
N_i = number of input neurons.
N_o = number of output neurons.
N_s = number of samples in training data set.
α = an arbitrary scaling factor usually 2-10.

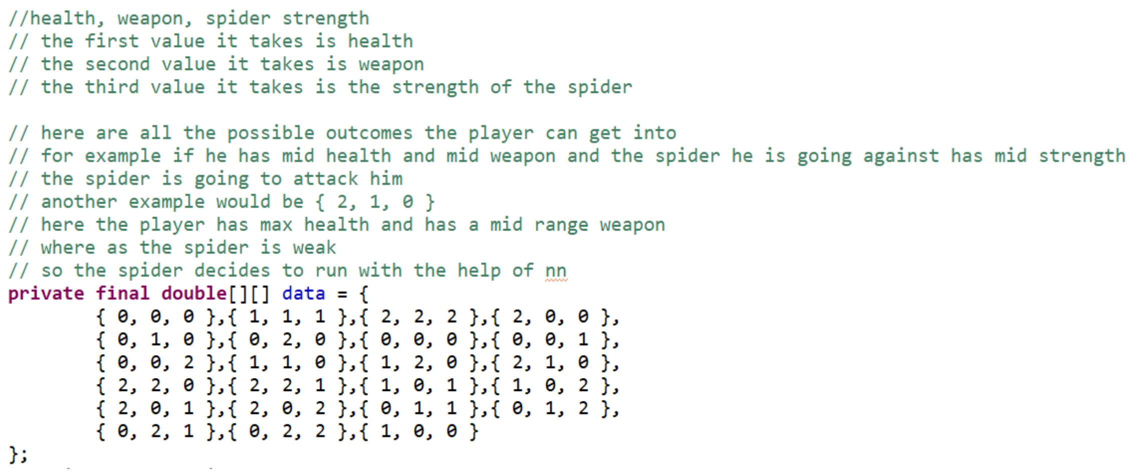
- The output layer returns the 3 neurons which are attack, panic and hide.

Training Data

I have included the data and expected data in a txt file in the resource folder explaining what each line does and what outcome should the neural network give.

Data

This is the data I use to train my neural network with.



```java
//health, weapon, spider strength
// the first value it takes is health
// the second value it takes is weapon
// the third value it takes is the strength of the spider

// here are all the possible outcomes the player can get into
// for example if he has mid health and mid weapon and the spider he is going against has mid strength
// the spider is going to attack him
// another example would be { 2, 1, 0 }
// here the player has max health and has a mid range weapon
// where as the spider is weak
// so the spider decides to run with the help of nn
private final double[][] data = {
        { 0, 0, 0 },{ 1, 1, 1 },{ 2, 2, 2 },{ 2, 0, 0 },
        { 0, 1, 0 },{ 0, 2, 0 },{ 0, 0, 0 },{ 0, 0, 1 },
        { 0, 0, 2 },{ 1, 1, 0 },{ 1, 2, 0 },{ 2, 1, 0 },
        { 2, 2, 0 },{ 2, 2, 1 },{ 1, 0, 1 },{ 1, 0, 2 },
        { 2, 0, 1 },{ 2, 0, 2 },{ 0, 1, 1 },{ 0, 1, 2 },
        { 0, 2, 1 },{ 0, 2, 2 },{ 1, 0, 0 }
};
```

Expected Data

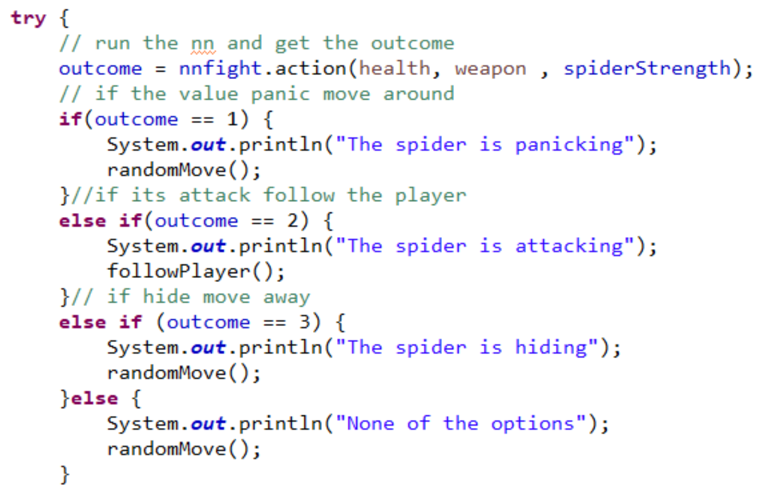
```
//Panic, Attack, Hide
private final double[][] expected = {

    //1) when the weak health + weak weapon + weak strength spider = attack
    //2) when the mid health + mid weapon + mid strength spider = attack
    //3) when the Max health + strong weapon + Strong strength spider = attack
    //4) when the Max health + weak weapon + weak spider strength = hide
    { 0.0, 1.0, 0.0 },{ 0.0, 1.0, 0.0 },{ 0.0, 1.0, 0.0 },{ 0.0, 0.0, 1.0 },

    //1) when the weak health + mid weapon + weak strength spider = panic
    //2) when the weak health + Strong weapon + weak strength spider = hide
    //3) when the weak health + weak weapon + weak strength spider = attack
    //4) when the weak health + weak weapon + weak spider strength = attack
    { 1.0, 0.0, 0.0 },{ 0.0, 0.0, 1.0 },{ 0.0, 1.0, 0.0 },{ 0.0, 1.0, 0.0 },

    //1) when the weak health + weak weapon + Strong strength spider = attack
    //2) when the mid health + mid weapon + weak strength spider = hide
    //3) when the mid health + strong weapon + weak strength spider = hide
    //4) when the strong health + mid weapon + weak spider strength = hide
    { 0.0, 1.0, 0.0 },{ 0.0, 0.0, 1.0 },{ 0.0, 0.0, 1.0 },{ 0.0, 0.0, 1.0 },

    //1) when the strong health + strong weapon + weak strength spider = hide
    //2) when the strong health + strong weapon + mid strength spider = panic
    //3) when the mid health + weak weapon + mid strength spider = panic
    //4) when the mid health + weak weapon + Strong strength spider = attack
    { 0.0, 0.0, 1.0 },{ 1.0, 0.0, 0.0 },{ 1.0, 0.0, 0.0 },{ 0.0, 1.0, 0.0 },

    //1) when the strong health + strong weapon + mid strength spider = panic
    //2) when the strong health + strong weapon + Strong strength spider = attack
    //3) when the weak health + mid weapon + weak spider = panic
    //4) when the weak health + mid weapon + strong spider = attack
    { 1.0, 0.0, 0.0 },{ 0.0, 1.0, 0.0 },{ 1.0, 0.0, 0.0 },{ 0.0, 1.0, 0.0 },

    //1) when the weak health + strong weapon + mid strength spider = panic
    //2) when the weak health + strong weapon + Strong strength spider = attack
    //3) when the mid health + weak weapon + weak strength spider = hide
    { 1.0, 0.0, 0.0 },{ 0.0, 1.0, 0.0 },{ 0.0, 0.0, 1.0 }
};
```

Training

```
// train the net
public void train() {
    nn = new NeuralNetwork(Activator.ActivationFunction.Sigmoid, 3, 3, 3);
    Trainator trainer = new BackpropagationTrainer(nn);
    trainer.train(data, expected, 0.2, 10000);
}

// this method returns what the neural net thinks the spider should do //O(N)
public int action(double health, double weapon, double angerLevel) throws Exception{

    //parameters
    double[] params = {health, weapon, angerLevel};
    double[] result = nn.process(params);

    // get the max result
    // value that is more likely out of the 3
    int choice = (Utils.getMaxIndex(result) + 1);

    // print out the prediction values
    for(double val : result){
        System.out.println(val);
    }

    // return the choice
    return choice;
}
```

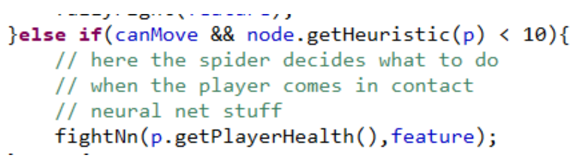
Game Integration

I have a method that is being called inside the run method. This method here calls the action method from up above and decides what to do. If the outcome of the method is 2 then the spider will attack and follow the player. When the player picks the one of the weapons his current weapon is set to that. so, when the spider uses the Neural Network it uses the players current weapon and players current health and the strength of the spider.



```
try {
    // run the nn and get the outcome
    outcome = nnfight.action(health, weapon , spiderStrength);
    // if the value panic move around
    if(outcome == 1) {
        System.out.println("The spider is panicking");
        randomMove();
    }//if its attack follow the player
    else if(outcome == 2) {
        System.out.println("The spider is attacking");
        followPlayer();
    }// if hide move away
    else if (outcome == 3) {
        System.out.println("The spider is hiding");
        randomMove();
    }else {
        System.out.println("None of the options");
        randomMove();
    }
```

I check in my run method if the player is in the range of 10 nodes. If the player is in range, then I run the neural network method to find out what the spider should do.



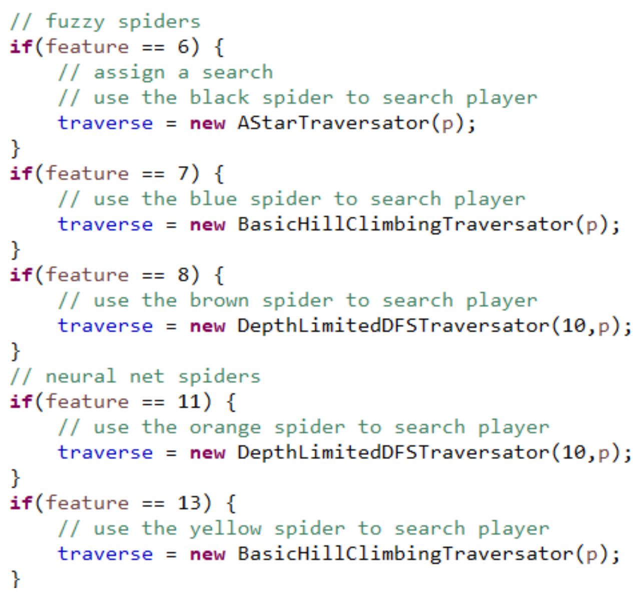
```
}else if(canMove && node.getHeuristic(p) < 10){
    // here the spider decides what to do
    // when the player comes in contact
    // neural net stuff
    fightNn(p.getPlayerHealth(),feature);
```

Heuristic Search

I use 5 different spiders to run different Heuristic searches. Spider nodes implement their own traverses to locate the player and to move either towards or away from the player.

I am using these 3 searches to find the player. The black, blue, brown, orange and yellow spiders are looking for the player.

- AStarTraversator
- BasicHillClimbingTraversator
- DepthLimitedDFSTraversator

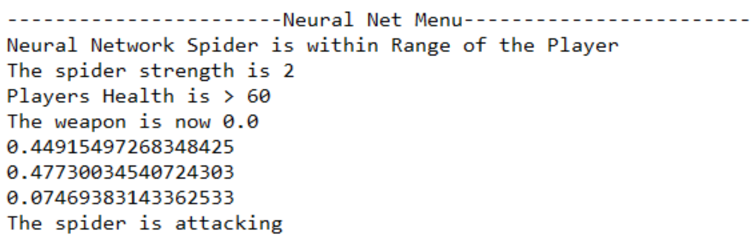


```
// fuzzy spiders
if(feature == 6) {
    // assign a search
    // use the black spider to search player
    traverse = new AStarTraversator(p);
}
if(feature == 7) {
    // use the blue spider to search player
    traverse = new BasicHillClimbingTraversator(p);
}
if(feature == 8) {
    // use the brown spider to search player
    traverse = new DepthLimitedDFSTraversator(10,p);
}
// neural net spiders
if(feature == 11) {
    // use the orange spider to search player
    traverse = new DepthLimitedDFSTraversator(10,p);
}
if(feature == 13) {
    // use the yellow spider to search player
    traverse = new BasicHillClimbingTraversator(p);
}
```

When the player is 10 nodes away from the spider the spider will start following the player and attack him. The spider will also run the neural network to see if it should really attack the player. Not all the searches are perfect. I use AStarTraversator which is highly accurate works perfectly, but others are not very accurate for this game.

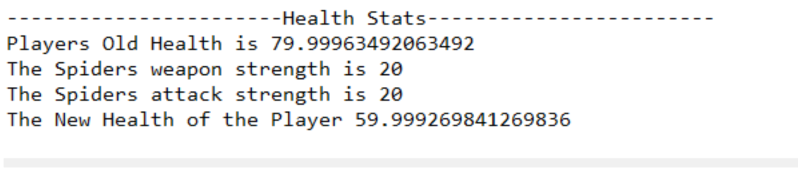
Everything together

- When the Spider is near the player (10 nodes away) the spider will run the neural network to decide what to do. For example



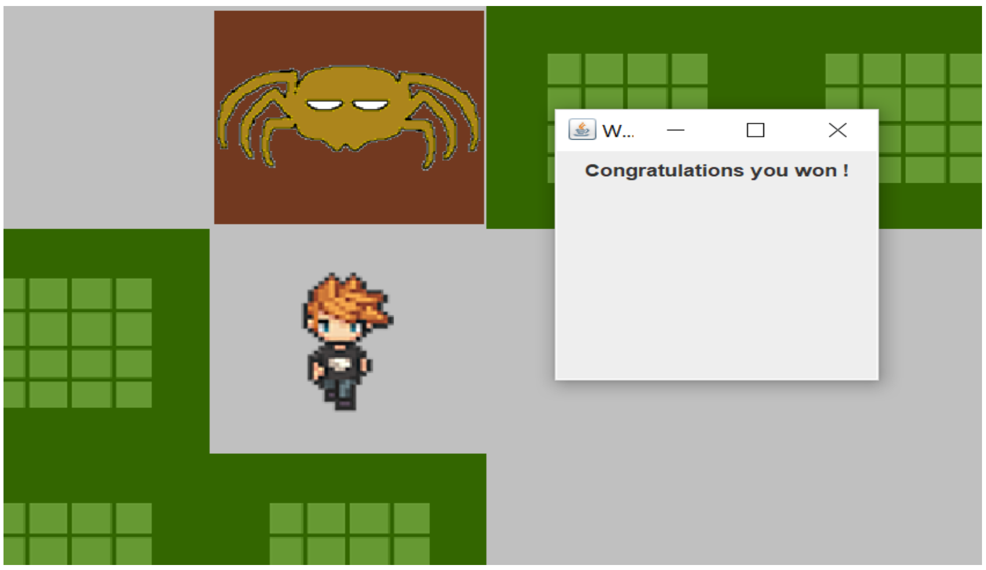
```
-----------------------Neural Net Menu-----------------------
Neural Network Spider is within Range of the Player
The spider strength is 2
Players Health is > 60
The weapon is now 0.0
0.44915497268348425
0.47730034540724303
0.07469383143362533
The spider is attacking
```

- When the Spider decides to attack the player, the fuzzy logic is used to give a new health to the player.

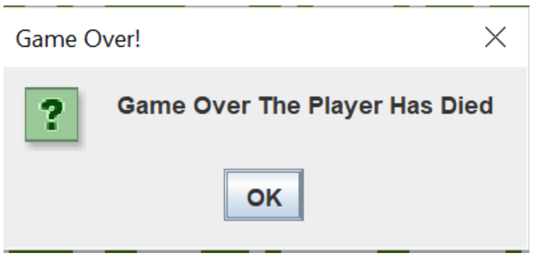


```
-----------------------Health Stats-----------------------
Players Old Health is 79.99963492063492
The Spiders weapon strength is 20
The Spiders attack strength is 20
The New Health of the Player 59.999269841269836
```

- The game is won when the player finds the door which is the colour pink on the map.



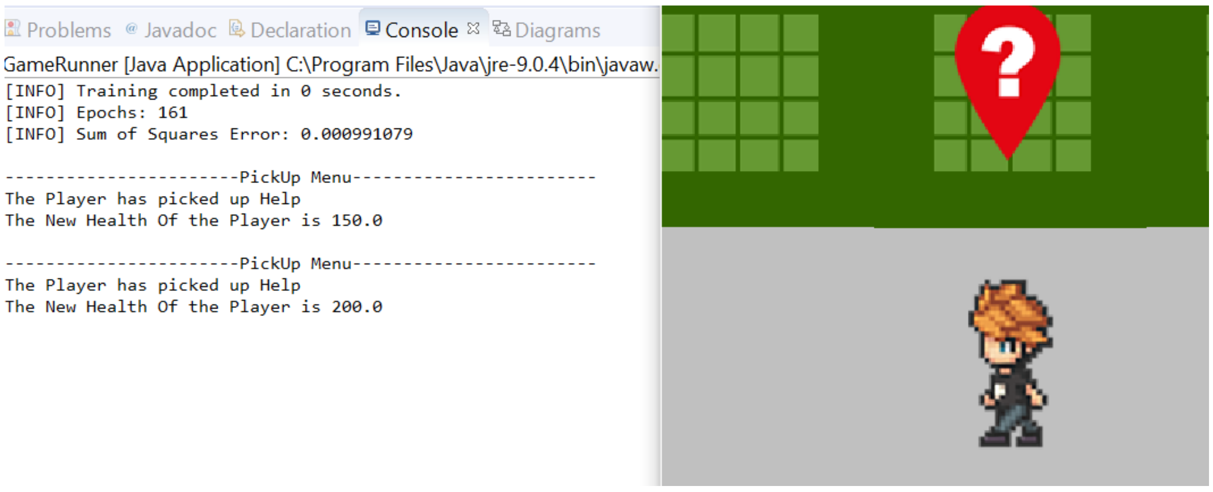
- when the spiders find the player, they try to kill him, when they do the game ends



Game Over!

Game Over The Player Has Died

- Also, the player can increase its health by picking up items



```
GameRunner [Java Application] C:\Program Files\Java\jre-9.0.4\bin\javaw.
[INFO] Training completed in 0 seconds.
[INFO] Epochs: 161
[INFO] Sum of Squares Error: 0.000991079

----------------------PickUp Menu-----------------------
The Player has picked up Help
The New Health Of the Player is 150.0

----------------------PickUp Menu-----------------------
The Player has picked up Help
The New Health Of the Player is 200.0
```

- The run method is being used to execute the spiders and player on separate threads

Spider Damage

spider	weapon,	enemy	=	damage	Type of Traverse?	Neural Network(panic, attack, hide)
Black spider	70	80	=	65	AStarTraversator	Determines What to do when player is near
Blue spider	65	65	=	60.1	BasicHillClimbingTraversator	Determines What to do when player is near
Brown spider	55	60	=	44.1	DepthLimitedDFSTraversator	Determines What to do when player is near
Green spider	45	50	=	38.6	Will Move Randomly	Moves Randomly
Grey spider	40	40	=	36.3	Will Move Randomly	Moves Randomly
Orange spider	70	80	=	65	DepthLimitedDFSTraversator	Determines What to do when player is near
Red spider	30	20	=	30	Will Move Randomly	Moves Randomly
Yellow spider	20	20	=	20	BasicHillClimbingTraversator	Determines What to do when player is near

Extras

- I made the maze into node [][] maze.
- The player can pick up the items to gain more health.
- I coloured all the spiders when z is pressed you can see all the coloured spiders
- only displaying 11 spiders who search for the player
- The player can also exit the game by finding the door which is pink.
- Redesign AStar traverser for it to be compatible with the maze and other searches.
- if the player loses health and is below zero the game automatically ends.
- Threads are designed to avoid deadlock
- The big O Notation added