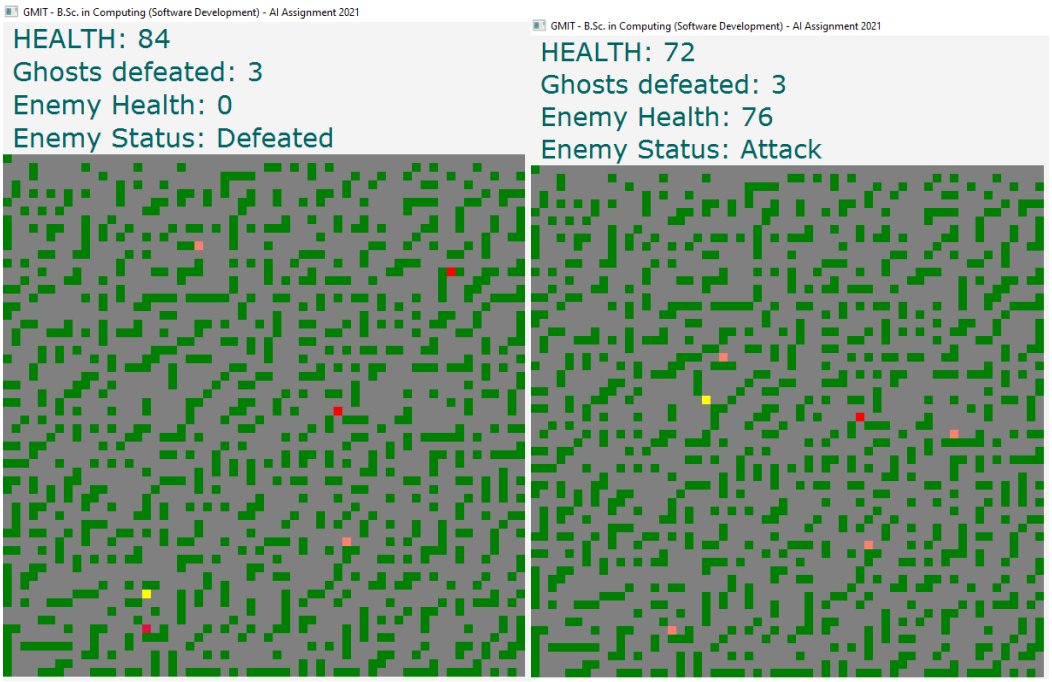
Autonomous Computer-controlled Game Characters

G00359994



Screenshots from the game.

Run on full screen

Run Application

java --module-path .;/path/to/your/javafx-sdk11.0.2/lib;/path/to/your/encog-3.4/;/path/to/your/jfuzzylogic --module gmit.software/ie.gmit.sw.ai.Runner

CharacterTask.java

- This class uses an instance of Enemy, Player and GameController.
- In the while loop the GameController handles all the logic of the game.

```
GC.playerRange(player.getRow(), row, player.getCol(), col);
GC.IsEnemyAlive(enemy);
GC.controlGame(player, enemy);
GC.IsPlayerAlive(player);

if (GC.IsEnemyAlive(enemy) == false) {
    player.setGhostsDefeated(1);
    player.PlayerRegen();
    System.out.println("Ghost " + enemyID + " defeated!!");
    alive = false;
    GC.HasplayerWon(player);
}
```

- Having a separate class that handles all the logic of the application increases encapsulation.

Enemy.java

- This class holds the getters/setters for the ghost's variables e.g health and strength.
- This class also implements Command so we can pass an instance of enemy to character task.

FuzzyNetEnemy.java

- This class extends Enemy and implements Command, so it has the Execute method.
- In the execute method we call the fuzzy network and pass in the health and strength.

```
public void execute() {
    double output = 0;
    try {
        output = new LoadNetworks().Runfuzzy(health, strengthvalue);
        if (output >= 65) {
            Command = "Attack";

        } else if (output >= 35 && output < 65) {
            Command = "Roam";
        } else {
            Command = "Run";
        }
```

- The output of this class will assign a command to the fuzzy net enemies.

GameController.java

- This class contains the methods used for controlling the game.

```
public double playerRange(double x1 , double x2 ,double y1, double y2) {
    range = Math.sqrt((Math.pow((x1 - x2 ), 2)) + (Math.pow((y1 - y2), 2))); //distance between ghosts and player;
    return range;

}
```

- This method returns the distance between the player and every ghost.

```
public void controlGame(Player player , Enemy enemy) {


    if(range <= 1  && enemy.getCommand() == "Attack") {  // Enemy attack
        player.gotHit(4);
        System.out.println("My health "+ player.getHealth());
    }

    else if(enemy.getCommand() == "Roam") {
        System.out.println("Enemy is roaming");
        enemy.EnemyRegen();


    }
    else if(enemy.getCommand() == "Run") {
        System.out.println("Enemy is Running");

    }
```

- In the control game method, it decides what the player/enemy will do.
- If the enemy is roaming it will get health back and go into attacking.

LoadNetworks.java

- This class is responsible for loading in the networks and running the networks.

Neural Network

```
public void LoadNeural() throws ClassNotFoundException, IOException {
    try {
        Neuralnet = (NeuralNetwork) SerializeObject.load(new File("EnemyNetwork_Neural"));
```

- The neural net is loaded in as follows.

```
public int RunNeural(double[] data) throws Exception {
    double[] result = Neuralnet.process(data);
    int output;
    output = Utils.getMaxIndex(result) + 1;
    return output;
```

- This method is used to take in strength and health to return the command to an enemy.

Fuzzy Network

```
public void LoadFuzzy() throws ClassNotFoundException, IOException {
    fuzzyNet = FIS.load("src\\fcl\\FuzzyNet.fcl", true); // Load and parse the FCL
```

- The network is loaded in as follows.

```
public double Runfuzzy(int health, int strength) throws Exception {

    FunctionBlock fb = fuzzyNet.getFunctionBlock("FuzzyNetEnemy");
    fuzzyNet.setVariable("Health", health); // Apply a value to a variable
    fuzzyNet.setVariable("Strength", strength);
    fuzzyNet.evaluate(); // Execute the fuzzy inference engine
    double ouput = fuzzyNet.getVariable("Command").getValue();
    // System.out.println(fuzzyNet.getVariable("Command").getValue()); //Output end
    // result
    return ouput;
```

- This method is used to take in strength and health to return the command to an enemy.

NeuralNetEnemy.java

- This class extends Enemy and implements Command.
- Call the neural network and pass in the health and strength.

```java
@Override
public void execute() {
    if (health >= 75) {
        healthStatus = 2;
    } else if (health >= 45 && health < 75) {
        healthStatus = 1;
    } else {
        healthStatus = 0;
    }

    if (strength == "Strong") {
        strengthStatus = 2;
    } else if (strength == "Normal") {
        strengthStatus = 1;
    } else {
        strengthStatus = 0;
    }

    // Attack, Hide, Run
    double[] data = { healthStatus, strengthStatus };
    int output = 0;
    try {
        output = new LoadNetworks().RunNeural(data);
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    if (output == 1) {
        Command = "Attack";

    } else if (output == 2) {
        Command = "Roam";
    } else {
        Command = "Run";
    }
```

- The output will decide the command for the ghost.

Player.java

- This class holds the getters/setters for the player variables e.g health, position and ghosts defeated counter.

Extras

- Display data on game window.