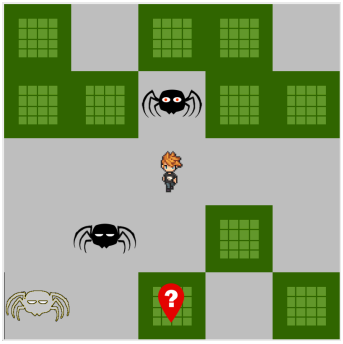
GALWAY-MAYO INSTITUTE OF TECHNOLOGY

Department of Computer Science & Applied Physics

B.Sc. Software Development – Artificial Intelligence (2019)
ASSIGNMENT DESCRIPTION & SCHEDULE

Controlling Game Characters with a Neural Network and Fuzzy Logic



Note: This assignment constitutes 50% of the total marks for this module.

You are required to create an AI controlled maze game containing the set of features listed below using a suite of stubs provided on Moodle. The objective of the game should be to escape from a maze and either avoid or fight off the enemy characters that move around in the game environment. The provided stubs already create a basic maze game, with a key-controlled player and (immobile) sprite enemies. Several game objects, such as swords, help points and bombs, have already been added to the game.

The purpose of this assignment is not to programme a game, but to *design a fuzzy logic system and neural network(s) that can control the enemy characters* and make them act in a pseudo-intelligent manner. As such, much of the programming will be declarative (fuzzy control language) or secondary to AI design considerations, e.g. neural network topology and training data.

Minimum Requirements

The following features are required:

- The game should **generate maze with one or more exits**. The player character should be placed in the maze as far from the exit(s) as possible. The objective of the game should be to find a way out of the maze. The game should be **graphical** and use arrow keys and other options to move around.
- The maze should be patrolled by enemies (spiders are already provided) or other **characters that are fully threaded**, i.e. the creatures should move independently through the maze. Each character should be controlled by some intelligence, either fuzzy logic, a neural network, a heuristically informed search algorithm or even a cocktail of all three.
- The fuzzy logic should be encapsulated in a FCL file and placed in the ***./resources/fuzzy***

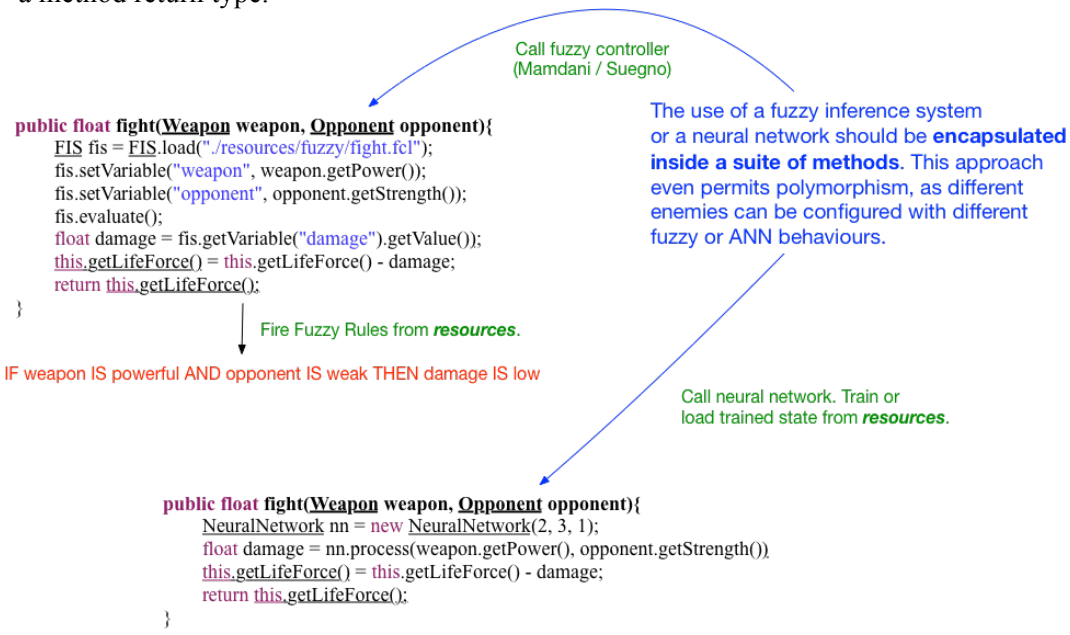
directory. You should consider carefully the following when designing the fuzzy logic components of the system:

1. **Linguistic variables** and Universe of Discourse
2. **Fuzzy Sets** and Membership Functions
3. **Defuzzifiers**, e.g. COG, COGS, LM, MM, RM.
4. Membership Functions for output variables, e.g. **Mamdani** or **Suegno** (singleton values)
5. **Fuzzy Rules**. All fuzzy sets must be covered by at least one fuzzy rule.

You must ***comment the FCL with your rationale*** for each of the design decisions that you make. Note that you can include more than one function block per FCL file.

- The neural network should be trained, in a reasonable amount of time (< 1 minute), when the game starts up and have a **fully documented topology** presented in the README. You are free to use the either the 3-layer neural network from Moodle or the ***Encog*** neural network that is also available on Moodle. Do not use any other 3rd party implementation. All the training and validation resources for the neural network should be placed in the ***./resources/neural*** directory.

You should view the application of the fuzzy logic or neural network as an "intelligent" implementation of a standard method. Consider the example method ***fight()*** shown below. The method parameters can easily be used as the input to either a fuzzy inference system or a neural network. The output from the fuzzy controller or neural network can be used as a method return type.



Note: you are free to asset-strip any of the resources, including labs and source code, available on the Moodle page for this module.

Deployment and Submission

- ***The project must be submitted by midnight on Sunday 21st April 2019*** as a Zip archive ***(not a rar or WinRar file)*** using the Moodle upload utility. You can find the area to upload the project under the *"Controlling Game Characters with a Neural Network and Fuzzy Logic - (50%) Assignment Upload"* heading in the *"Main Assignment"* section of Moodle.

- The name of the Zip archive should be *{id}*.zip where *{id}* is your student number.
- You must use the package name **ie.gmit.sw.ai** for the assignment.
- Do not include any additional JAR archives with your submission. You can assume that both *jFuzzyLogic.jar* and *encog-core-3.4.jar* are already on the CLASSAPATH.
- The Zip archive should have the following structure (do NOT submit the assignment as an Eclipse project):

Name	Description
src	A directory that contains the packaged source code for your application. Package your application with the namespace **ie.gmit.sw.ai**. Your code should be well commented and explain the space and time complexity of its classes.
resources	A directory that contains **all the resources** required by the application. You can add to the existing set of subdirectories (*images, fuzzy, neural*) if necessary. All fuzzy definitions should be stored in the ***fuzzy*** directory and neural network training and validation data in the ***neural*** directory.
README.txt	A text file outlining the main **features** of your application. Marks will only be given for extra features that are described in the README.
game.jar	A Java Application Archive containing the classes in your application. Use the following syntax to create a JAR from a command prompt: **jar –cf game.jar ie/gmit/sw/ai/*.class** Your application must be able to be launched using the following syntax from a command prompt: **java –cp ./game.jar ie.gmit.sw.ai.GameRunner** This requires that the *main()* method for your application is in a class called *GameRunner*.

Marking Rubric

Marks for the project will be applied using the following criteria:

Marks	Category
(10%)	**Packaging & distribution.** All or nothing. The application must be structured correctly and execute without any manual intervention as described above.
(10%)	**Threads.** The player and enemy sprites should execute in separate threads.
(30%)	**Fuzzy Logic (10% for Each of the following):** • Fuzzy sets and membership functions with comments. • Fuzzy rules (all sets should be covered by at least one rule) with comments • Integration with game characters.
(30%)	**Neural Network (10% for Each of the following):** • Network topology, normalization and accompanying rationale. • Training and validation data. • Integration with game characters.
(20%)	**Heuristic Search (10% for Each of the following):** • Fully commented design in the README and implementation of heuristics for an AI search of the game space. • Implementation of algorithm(s) and integration with game characters.

Each of the categories above will be scored using the following criteria:

- 0–39% Not delivering on basic expectations
- 40-59% Meeting basic expectations
- 60–79% Tending to exceed expectations
- 80-90% Exceeding expectations
- 90-100% Exemplary