CSG2341 Intelligent Systems Assignment

Title: Saucers – Part A

Related objectives from the unit outline:

- design and use computer packages incorporating the techniques associated with computational intelligence;
- investigate the application of these intelligent systems techniques to a relevant problem;
- apply relevant technique(s) to a task normally considered to require computational intelligence.

Specific learning outcomes:

After completing this assignment, you should be able to apply intelligent systems techniques to design and implement a solution to a “realistic” problem.

Marks allocation:

This assignment is worth 40% of the total mark for this unit. The assignment is in two parts, with two submission dates. Part A, worth 15%, must be done in groups of 1, 2, or 3 people. Part B, worth 25%, will be an extension of Part A.

Due dates:

Part A: Midnight Sunday 30th August

Part B: Midnight Sunday 18th October

Referencing:

All sources of references must be cited (in text citation) and listed (end reference list). For details about referencing and the required format, please refer to the ECU Referencing Guide, which can be found from the following URL:
http://www.ecu.edu.au/LDS/pdf/refguide.pdf

Plagiarism:

Please ensure that you have read and understood the information on plagiarism provided on BlackBoard under Unit Overview->Essential Information.

Submission requirements:

Assignment materials are to be handed in by the due date, in electronic form by Blackboard submission.

Part A: (15 marks)

One of the application areas for Intelligent Systems methods is computer games – in particular, providing "intelligent" opponents for human players. In first-person shooters, these intelligent opponents are sometimes called "bots". In this assignment, you will use develop a fuzzy control program for a bot in a game called Saucers.

Saucers is a game for two players, each controlling an armed vehicle called a saucer, so called because of their appearance. The two saucers meet on a 2-dimensional rectangular battlefield. Each saucer is given a fixed amount of "energy" at the start of the game, and the first one to run out of energy is the loser. The saucers constantly move (fly?) over the battlefield, using up energy as they go. They can't stop, but they can be steered. However, turning a saucer also uses up energy. A saucer that runs into one of the walls of the battlefield will ricochet off it like a billiard ball. Saucers have a simple sensor system that tells them where the opposing saucer is and how much energy it has. Saucers can also "shoot" blasts of energy at their opponent, using self-aiming photon blasters. Each blast takes some energy from the saucer that shoots it, but can take more energy from the opposing saucer if it hits (depending how far away it is). Blasts travel in straight lines and cannot be steered, so if the opposing saucer changes direction after the shot is made, a blast will often miss. Also, blasts lose energy as they travel, eventually petering out if they don't hit anything. Saucers cannot detect incoming blasts.

There are many possible controller strategies for playing Saucers. Two examples, `SimpleController` and `FuzzyController` have been provided for you (with Java source code). Your task is to write a Java class for a controller to compete against these sample controllers, and another set of unseen controllers, using a set of fuzzy rules that you design. `SimpleController` does not use any fuzzy reasoning. `FuzzyController` shows you how you can include fuzzy rules in your controller.

Task:

You are to write a Java class that implements the interface SaucerController:

```
public interface SaucerController
{
    public void sensorUpdate
    (
        double opponentDistance,
        double opponentDirection,
        double opponentEnergy,
        double energy
    ) throws Exception;
```

```
    public double getFirePower() throws Exception;
    public double getTurn() throws Exception; // in degrees
    public double getSpeed() throws Exception;
    public String getName();
    public Color getBaseColor();
    public Color getTurretColor();
}
```

N.B. Your class must have a constructor with no parameters.

When the games starts, your controller's `sensorUpdate` method will be called, and it will be passed the distance between you and your opponent, the direction that your opponent is in, relative to the direction that you are currently moving (in degrees), how much energy your opponent has left (you both start with 10,000 units) and how much energy you have left. This gives you a chance to save any data values that you need and do any calculations that you want to do, based on this information. Then your controller's `getFirePower` method will be called to determine how strong a blast to shoot at your opponent (return 0.0 if you don't want to shoot). Note that the photon blaster cannot be used again for a short time after a blast – the blaster won't fire if you try before that, and no energy will be used. Then your controller's `getTurn` method will be called to determine how much to change your direction (in degrees). Positive values mean turning to the left, negative values mean turning to the right, 0.0 means keep going in the same direction. If you return the value `opponentDirection` that was passed to `sensorUpdate`, you will turn to face your opponent. Then your controller's `setSpeed` method will be called to determine how fast you want to go (going faster uses up energy faster). Note that there is a minimum and maximum allowed speed.

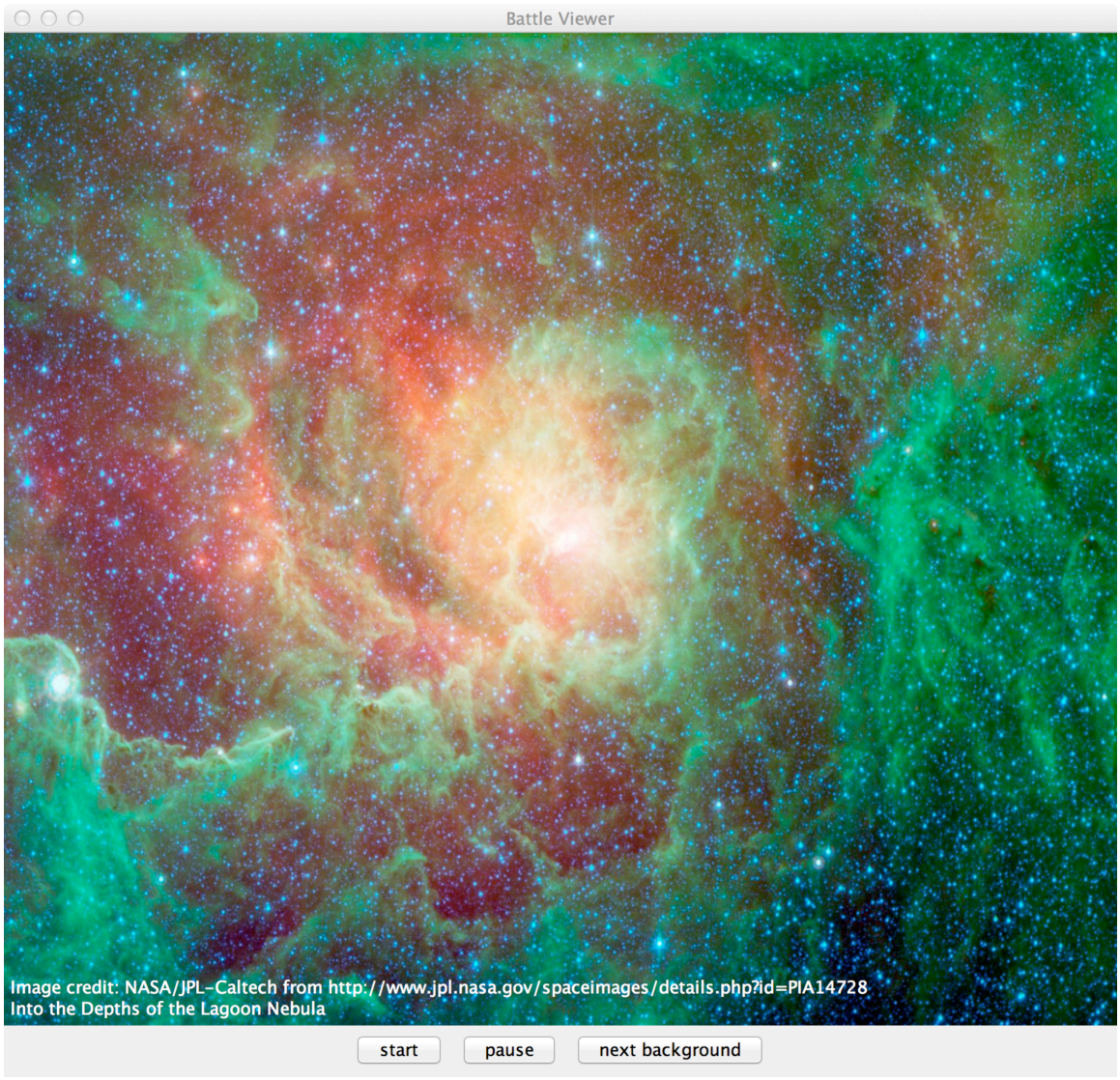
This sequence of calls will then be repeated over and over until the contest is decided (somebody runs out of energy).

The other methods, `getName`, `getBaseColor` and `getTurretColor` are used when drawing your saucer.

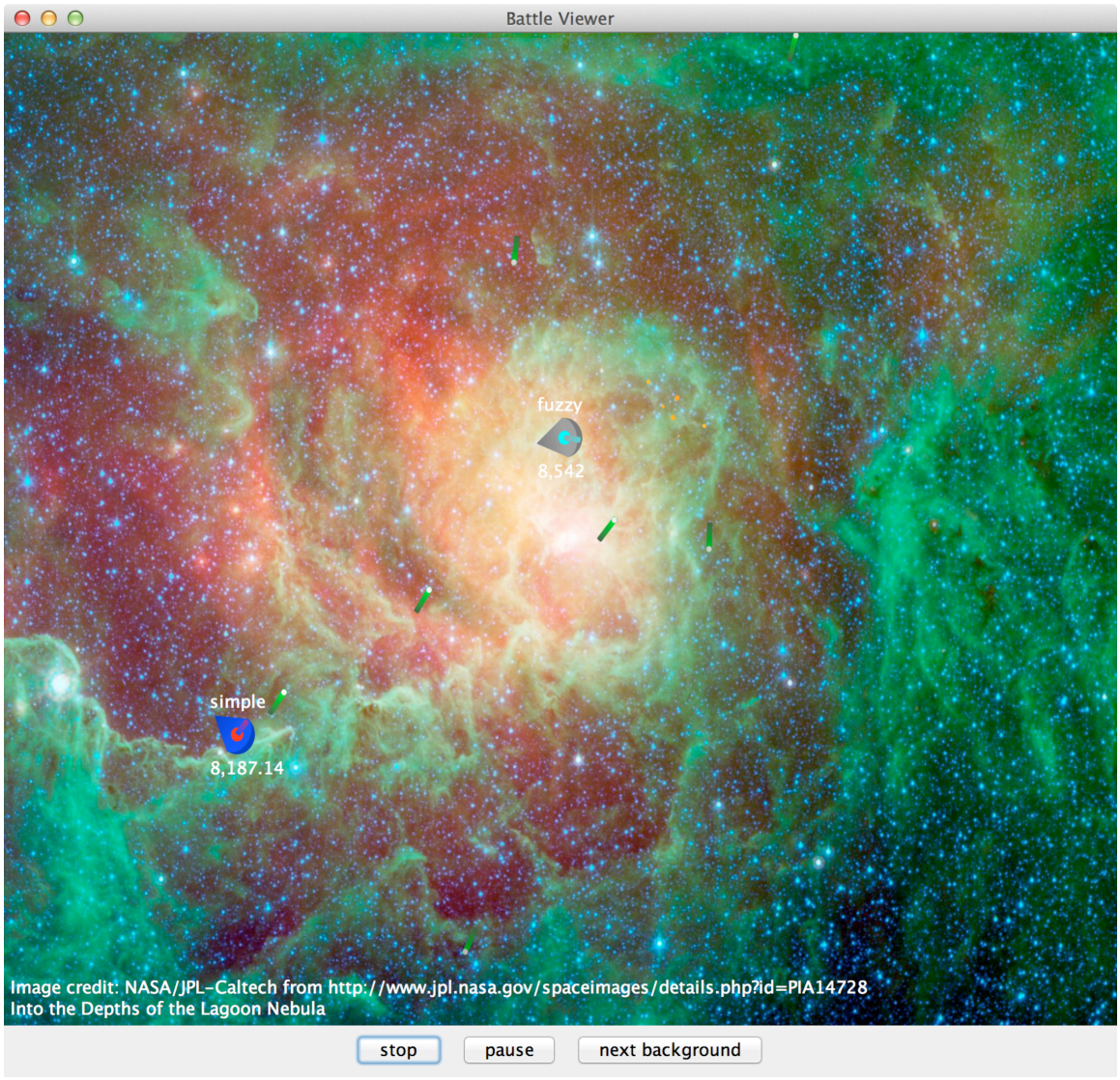
Instructions:

The files that you will need will be provided to you on BlackBoard, in the file `SaucersA.zip`. Extract the files from this archive into a suitable folder. One of its sub-folders is called `src` (if you do not have subfolders, then you have made a mistake extracting the archive). Use the folder as the project folder for a new NetBeans project, and use `src` as the source folder. If you do this correctly you will have a project with a number of related Java packages. Build the project, and run the `Main` class.

You should see a display like this:



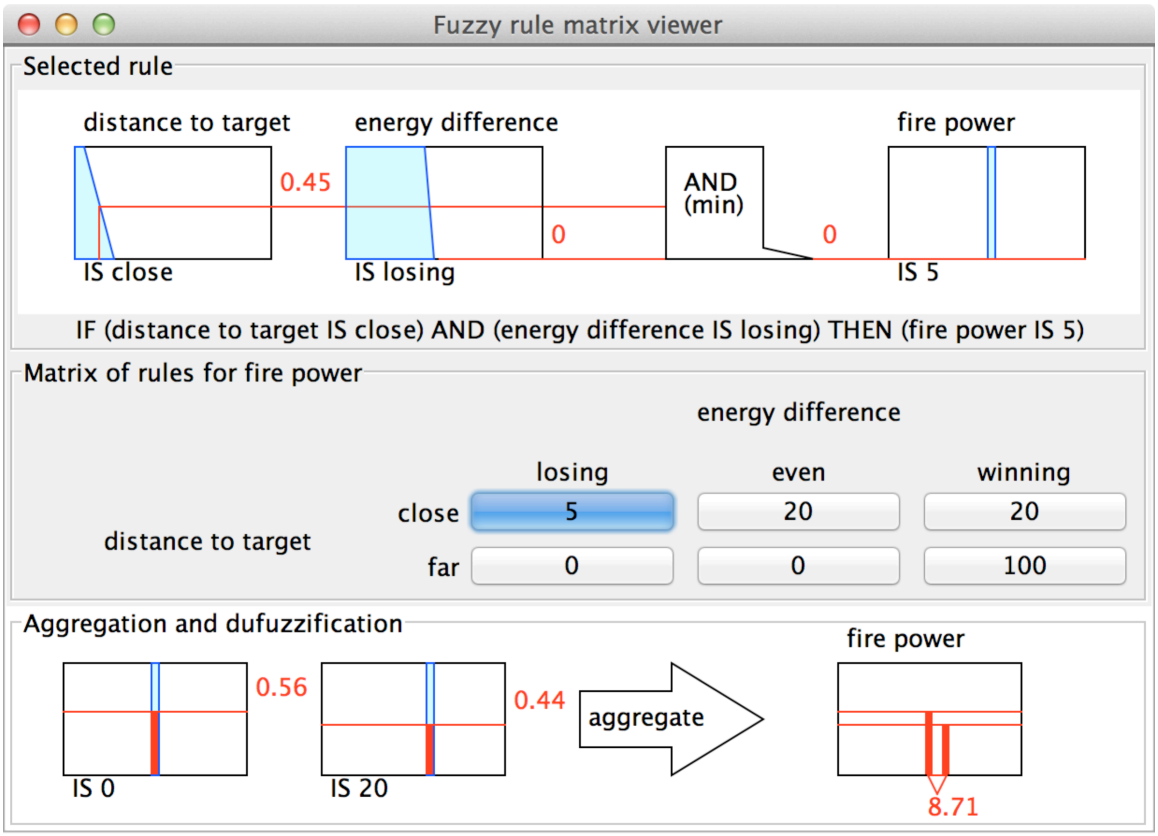
Click on “start” and you should see something like this:



The display shows two saucers, one using the `SimpleController` (you can look at the source for this one in `SimpleController.java`) and one using the `FuzzyController`. Simple has a blue base and a red turret. The pointy prow of the saucer points in the direction the saucer is heading. Simple has 8,187.14 units of energy remaining. You will some green streaks – these are photon blasts. The battle will continue until someone runs out of energy. The program will print out how much energy each saucer had at the end of the battle.

The program allows you to start and stop contests, and to pause and restart them, which could be useful when you are debugging and fine-tuning your controller.

You will also see a window like this:



This window has been created by `FuzzyController` and can be used to see which rules are firing. You can use the pause/resume button in the main window to stop the action and examine the fuzzy rules in more detail.

Your task is to write and test a Java class that implements the `SaucerController` interface. Your controller must use fuzzy rules.

Make up your own unique name for your controller class, and add it to the `battle.saucers.controllers` folder. Locate the following lines in `Main.java`:

```
SaucerController fuzzy = new FuzzyController();
    Saucer saucer1 = new Saucer(field, simple);
    Saucer saucer2 = new Saucer(field, fuzzy);
```

Add in your new controller here. To run lots of battles without the graphics and sound, you can use `Tournament.java`. You will need to make similar adjustments to the source code to add in your new controller.

This is the only change you are allowed to make to the existing source code. **You cannot change the existing classes in any way.**

It is suggested that your class should use a fuzzy rule set like that for the inverted pendulum and the cruise control that we have developed in lectures and workshops. You could start by writing down suitable rules in English, like

"If you are winning and your opponent is far away, don't fire, and stay far away"

Then try to figure out what fuzzy variables you would need to express these rules.

You will find various useful constants that can (should) be used rather than hard-coding values in your code. For example:

Constants.STARFIELD_WIDTH	This is the width of the battle area
Constants.STARFIELD_HEIGHT	This is its height
Saucer.START_ENERGY	This is how much energy the saucers start with
Saucer.MAX_POWER	This is the largest blast possible
Saucer.MIN_SPEED	This is the slowest a saucer can go
Saucer.MAX_SPEED	The is the fastest a saucer can go

Report

Write a 1200 word report documenting your work, with at least the following headings:

1. Introduction
   A short introduction. You don't need to provide details of the task – just refer to this document.
2. Idea
   Describe the idea behind your controller in simple English.
3. Fuzzy Variables
   List the input and output (and any other) fuzzy variables and include membership diagrams for each fuzzy set.
4. Example of a Fuzzy Rule
   Choose one fuzzy rule and use diagrams to illustrate how the firing strength and output variable value are calculated.
5. Learnings
   How well did your controller work, and what did you learn from Part A?
6. Conclusion
   A simple conclusion should round out the report. Summarise what you did and what you learned.

1200 words is a guide. The report doesn't have to be exactly 1200 words, but if it's a lot less, say 700 or 800, or if important information is missing, expect to lose marks.

Please include details of your team members (names and student numbers).

Marking Key:

Part A is worth 15% of your mark for Intelligent Systems. **Note that your Controller must use fuzzy rules to earn these marks, and it must be substantially different from FuzzyController. Just changing a few numbers is NOT substantially different.**

The marks are made up of

- Appropriate use of Intelligent Systems techniques
  Submit a zip file containing:
  - Your completed project report, and
  - Java source files for your controller. Just submit your WhateverYouCallIt.java – not the .class file, and not the rest of the source files (if you use additional classes, please put these into a Java package with a unique name, zip up the folder, and submit that too).

  (5 %)
- Performance of the controller (5% + 5%)

  Another 5% will be calculated using the formula: marks = 5*average score/1000, where the average score will be calculated by your tutor by running Tournament using your controller against the non-fuzzy controller, SimpleController. The minimum mark for this part is 0%, and the maximum is 5%.

  The final 5% will be calculated in the same way, from a tournament using your controller against two other “unseen” controllers.

  (10 %)

PLEASE READ THE NOTES ON THE NEXT PAGE CAREFULLY

Checklist of important things to remember for this assignment:

- You must work in groups of 1, 2 or 3.
- Only one submission for each group – be sure to include a list of group members in your report.
- **DO NOT** come along near the submission date and say "My group members haven't done their work/have gone home/annoy me - what should I do?." It is your responsibility to ensure that your group submits by the due date.
- If your group is having trouble getting started, get help from your tutor in plenty of time.
- If you are working from home, you will need to install a recent Java SDK, and the IS library (is.jar). There are some instructions in Workshop 2.
- Your task is to write a better "controller" for a saucer.
- Your "controller" **MUST** use fuzzy reasoning in a non-trivial way.
- Your "controller" **MUST** be significantly different from FuzzyController.java.
- You must hand in a zip file containing
    - Your project report,
    - Your source code.